Université de Toulouse

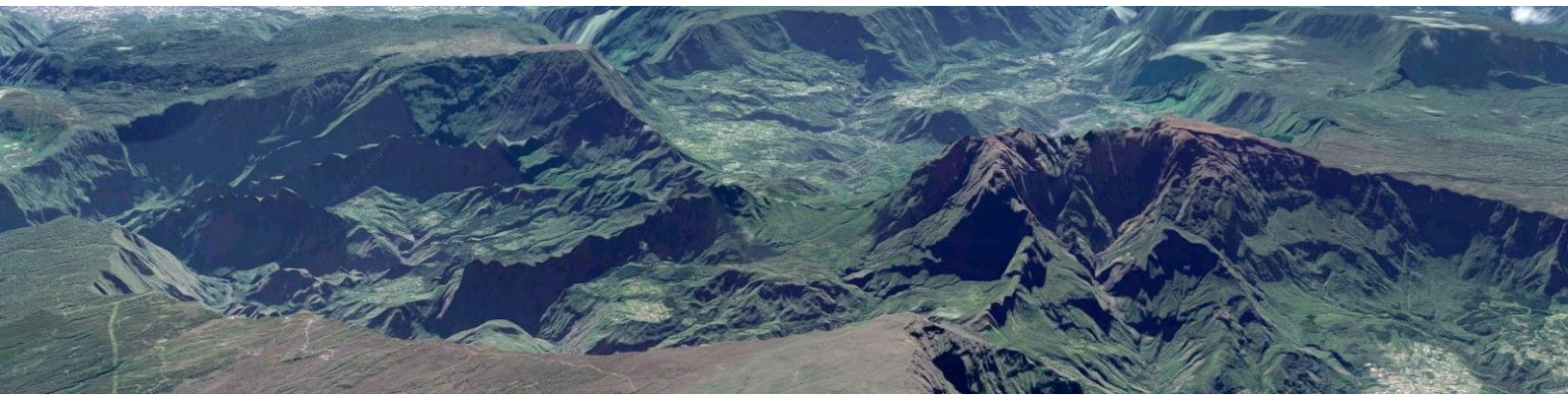
MASTER 2 GEOMATIQUE

« **S**cience de l'**I**nformation **G**éoréférencée pour la **M**aîtrise de l'environnement et l'**A**ménagement des territoires » (**SIGMA**)

http://sigma.univ-toulouse.fr

RAPPORT DE STAGE

Evolution d'un atlas cartographique web



ROLLAND Antoine

cirad

Centre de coopération internationale en recherche agronomique pour le développement

Maître de stage : Agnès TENDERO
Tuteur-enseignant : Laurent JÉGOU

Septembre 2016

Résumé

Au cours de l'année 2015, l'équipe ARTISTS du Cirad à La Réunion a mis en place un atlas cartographique web de données agro-environnementales baptisé *AWARE*, acronyme pour Atlas Web Agricole pour la REcherche (http://aware.cirad.fr). Cet atlas permet aux agents du Cirad d'avoir facilement accès à un catalogue de presque 300 couches, 17 cartes et 44 documents au 1er septembre 2016. Alimenté grâce aux contributeurs, il offre l'opportunité à la communauté de chercheurs de faire connaître les résultats de leurs travaux aux autres agents du Cirad mais aussi à un panel plus étendu d'utilisateurs. Il permet également de s'affranchir du problème des nombreuses données réparties sur des postes personnels, ou stockées sur les serveurs de la DSI (Direction des Systèmes d'Information) – jusque-là accessibles seulement sur demande. Aujourd'hui, chaque agent du Cirad a la possibilité de se connecter facilement sur la plateforme pour profiter de l'intégralité du catalogue de ressources (couches, cartes et documents).

Le but de ce stage est de développer de nouvelles fonctionnalités sur cette plateforme. Celle-ci est en effet basée sur des solutions technologiques libres (*GeoNode*/*GeoServer*) et donc personnalisable. Certaines fonctionnalités propres à AWARE modifiant le comportement de GeoNode ont déjà été développées, mais l'équipe souhaite maintenant effectuer une migration de la version 2.4 *pre-release* utilisée actuellement vers la version 2.4 stable proposée depuis novembre 2015 par GeoNode. Suite à cette migration, l'équipe voudrait améliorer la confidentialité pour les contributeurs – certains ont exprimé le souhait de pouvoir bénéficier de plus de confidentialité pour leurs ressources – et continuer à modifier le code source de la plateforme jusqu'à obtenir le comportement qu'ils attendent. Grâce à la réactivité des développeurs de GeoNode et la riche documentation disponible sur les technologies libres utilisées, il a été possible durant ce stage de faire évoluer ce site dans cette optique.

Abstract

During 2015, ARTISTS Team from Cirad in Reunion Island implemented a cartographic web atlas called AWARE, acronym for Agricultural Web Atlas for REsearch (aware.cirad.fr). This atlas allows Cirad agents to have an access to almost 300 layers, 17 maps and 44 documents. Feeded by contributors, it allows the community of researchers to broadcast the results of their work to other Cirad agents, but also to a wider range of users. It enables to solve the problem of unshared resources, which are often kept in personal computers, or which are stocked in DSI (Direction of Information Systems) servers – only available by request until now. Today, each Cirad agent can easily connect to the platform in order to benefit from the whole catalogue resources (layers, maps and documents).

The team now would like to develop new functionalities on this platform. Indeed, it is based on open source technological solutions (GeoNode/GeoServer), which are customizable. Part of specific AWARE functionalities have already been developed, modifying the original behavior of GeoNode, but the team would now particularly wish to migrate from 2.4 pre-release version, currently used, to stable 2.4 version, released in November 2015 by GeoNode. Following that, they wish to improve the confidentiality – some users requested a better confidentiality for their resources to the team – and keep shaping the platform until it fits the behavior they aim. Thanks to GeoNode developers' responsiveness, and an extensive documentation concerning open solutions used in GeoNode, it was possible to make the website evolve in that way during this internship.

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage, et qui m'ont aidé lors de la rédaction de ce rapport.

Je tiens tout d'abord à remercier le Cirad pour m'avoir donné l'opportunité d'effectuer ce stage, dans ce cadre de travail tout simplement exceptionnel qu'est La Réunion.

Je remercie toute l'équipe ARTISTS, particulièrement Agnès Tendero pour sa grande disponibilité, son aide et sa bienveillance, que ce soit dans le cadre du stage ou en dehors, et Mickaël Mézino, pour la confiance qu'il m'a accordée, sa sérénité lors de certaines mésaventures avec le serveur et toutes ces heures qu'il m'a accordées pour m'aider dans le développement d'AWARE. Je les remercie sincèrement tous les deux pour avoir cru en mes capacités dès le début de ce stage, que j'abordais avec une certaine appréhension. Je les remercie enfin pour leur encadrement de qualité, et pour le temps qu'ils ont passé à m'aider dans la rédaction de mon rapport. Je remercie Pierre Todoroff pour m'avoir accueilli au sein de son équipe, et pour sa très agréable compagnie au quotidien. Je remercie enfin Lionel Le Mézo pour les moments sympathiques que j'ai eu l'occasion de passer avec lui durant mon stage.

Je remercie également Pascal Degenne, qui m'a aidé à mieux comprendre le fonctionnement de Git et GitHub, Vincent Bonnal, qui m'a éclairé sur le fonctionnement des traductions sur GeoNode, Raymond Nativel, ainsi que tous les stagiaires et VSC qui m'ont entouré lors de ce stage : Magali Jameux, Blandine Rosiès, Mathilde Stern, Boris Vaitilingon, Bastien Barral, Louise Randrianarivony ainsi que tous ceux du Cirad Ligne Paradis, sans qui mon séjour sur l'île n'aurait pas été le même.

Je tiens à remercier mon tuteur, Laurent Jégou, pour son aide dans ma recherche de stage, sa disponibilité, son intérêt pour mon travail et ses messages encourageants. Je remercie toute l'équipe du Master SIGMA pour leur enseignement de qualité, qui m'a permis d'acquérir de solides connaissances dans le domaine de la géomatique. Merci également l'Université Toulouse Jean Jaurès pour cette très agréable année que j'y ai passée.

Enfin, je remercie l'équipe de développement de GeoNode pour leur réactivité, et pour l'incroyable outil qu'ils ont développé, en espérant avoir pu apporter ma pierre à l'édifice.

Table des matières

I. Introduction

À La Réunion, l'équipe ARTISTS (« Artists of Remote sensing Tools, Information systems, Simulation Techniques and Spatial analysis »), équipe de recherche au sein du Cirad, a mis en place un Atlas cartographique web baptisé AWARE. Il permet aux chercheurs de publier et de partager les résultats cartographiques de leurs recherches, mais également de prendre connaissances des travaux des autres chercheurs, et de bénéficier des ressources des partenaires du Cirad, tels que l'IGN. C'est également un moyen pour eux de stocker leurs données sur une plateforme fiable, sécurisée et en ligne. Enfin, AWARE offre l'opportunité de créer et personnaliser des cartes à partir de couches existantes, sans utiliser un logiciel de SIG. À ce jour, ce catalogue compte presque 300 couches, 18 cartes et 44 documents.

AWARE est basé sur l'infrastructure de données spatiales (IDS) *GeoNode*, une solution technologique libre qui permet de le personnaliser et l'adapter pour répondre au cahier des charges de l'équipe. Si les chercheurs sont prêts à exposer leurs données sur la plateforme, ils demandent à ce que la confidentialité des données soit respectée. L'équipe ARTISTS a déjà réalisé de nombreuses modifications pour personnaliser le comportement d'AWARE par rapport à celui de GeoNode, mais elle souhaiterait maintenant renforcer la confidentialité sur la plateforme, tout en gardant une grande flexibilité pour ses utilisateurs. Pour cela, il sera nécessaire de modifier le comportement original prévu par GeoNode.

L'équipe ARTISTS a également mis en place ce stage afin d'assurer la migration d'AWARE, basé sur une version de développement (*pre-release*) de la 2.4 de GeoNode, vers la version 2.4 stable, sortie en novembre 2015.
Consciente que les modifications apportées pourraient intéresser certains utilisateurs ou futurs utilisateurs de GeoNode, l'équipe souhaiterait remonter sa version à la communauté, en soumettant fonctionnalité par fonctionnalité les spécificités d'AWARE. Si les développeurs de GeoNode intègrent ces fonctionnalités, il sera plus simple par la suite d'effectuer une migration vers les prochaines versions de cette IDS. L'équipe souhaiterait également rendre sa version téléchargeable et réutilisable telle quelle, pour les utilisateurs qui seraient à la recherche d'une IDS ayant le comportement d'AWARE.

Ce stage de fin de Master 2 étant mon stage de fin d'études, il représente une réelle transition entre le milieu universitaire et le milieu professionnel. Je ne l'ai donc pas choisi par hasard, il était très important pour moi que le sujet du stage et les missions proposées soient en accord avec mes ambitions. Ayant suivi une formation en géomatique ces deux dernières années de Master, j'avais particulièrement apprécié les projets de webmapping, et de développement web. Je souhaitais réaliser un stage qui me permettrait de développer mes compétences et d'aller plus loin dans ces domaines, et ce stage répondait à mes attentes en tout point. N'étant pas informaticien de formation, les missions proposées représentaient un réel challenge pour moi. Je l'ai débuté avec de l'appréhension, mais en étant certain qu'à son issue j'aurais acquis les compétences que je visais.

L'équipe ARTISTS m'a donc confié plusieurs missions. Tout d'abord améliorer la confidentialité sur la plateforme AWARE, effectuer la migration vers GeoNode 2.4 stable, puis remonter les spécificités d'AWARE à la communauté. Ces missions constituent l'axe principal du stage. Parallèlement, un certain nombre d'autres tâches sont à réaliser, toutes ayant pour but d'améliorer la plateforme, tant du point de vue fonctionnel qu'ergonomique et esthétique. Ces missions ont pu être menées à bien, grâce à une communauté de développeurs réactive, motivée et compétente, un environnement de travail bien encadré au sein de l'équipe, et une documentation assez fournie pour tous les langages et applications que j'ai utilisés.

II. Structure d'accueil

1. Le Cirad

Le Cirad (Centre de coopération Internationale en Recherche Agronomique pour le Développement) est un établissement public à caractère commercial (EPIC) français, créé en 1984. Il réalise des recherches en agronomie tropicale dans les domaines de protection des plantes, des risques environnementaux, de la gestion intégrée des ressources et de la qualité des productions agricoles et des produits alimentaires. Le siège social se situe à Paris, deux centres de recherche existent à Montpellier et Montferrier-sur-Lez. Il existe ensuite de nombreuses stations Outre-Mer et à l'étranger.

Fig. 1 - Les stations du Cirad en Métropole et dans le Monde. Source : Cirad

Le Cirad réunit environ 1650 personnes, dont 800 chercheurs. Il comprend 34 Unités de Recherche, réparties dans trois départements scientifiques : le département *Systèmes Biologiques* (**Bios**), le département *Performances des Systèmes de production* (**Persyst**) et le département *Environnement et Sociétés* (**ES**).

Fig. 2 - L'équipe ARTISTS au sein du Cirad – Organigramme simplifié

La mission du Cirad, en partenariat avec les pays du Sud, est de produire et transmettre de nouvelles connaissances, pour accompagner leur développement agricole et contribuer au débat sur les grands enjeux mondiaux de l'agronomie. Cette mission est légitimée par le contexte international : prégnance des grands défis mondiaux de l'humanité (maladies émergentes, gestion de la biodiversité, sécurisation des récoltes…), interdépendance des problématiques de recherche pour le développement, intérêt croissant pour les questions agricoles dans l'agenda du développement.

Les partenaires du Cirad sont nombreux. Pour n'en citer que quelques-uns : La Banque Mondiale, l'Organisation des Nations Unies pour l'Alimentation et l'Agriculture (FAO), le Fonds International pour le Développement Agricole (FIDA). Il participe au réseau Recherche Agronomique pour le Développement (RAD) de l'Espace européen de la recherche, qui finance des projets sur la sûreté des aliments, et le renforcement des compétences au Sud. Il est également engagé dans un partenariat scientifique international, reconnu par des accords bilatéraux et multilatéraux de plus de 90 pays.

2. Le Cirad à La Réunion et à Mayotte

La Réunion accueille le deuxième dispositif du Cirad en France, après Montpellier. Sa mission principale est de contribuer au développement des filières agricoles et agroalimentaires, en préservant l'environnement et sa biodiversité.

Il répond à trois missions :

- Produire des résultats scientifiques d'excellence,
- Répondre aux besoins de développement rural de l'île de La Réunion et de l'île Maurice,
- Mener une politique de coopération régionale active dans l'Océan Indien.

A La Réunion, les recherches sont programmées et financées pour six ans, de 2015 à 2020, dans le cadre d'un accord-cadre entre l'État, la Région Réunion, le Département de La Réunion et le Cirad. Les objectifs de l'intervention du Cirad à La Réunion sont en phase avec la stratégie bioéconomique de l'Union Européenne. Il s'agit de :

- Conforter le rôle de La Réunion comme plateforme européenne de recherche en milieu tropical,
- Renforcer l'articulation entre sciences et pratiques au service du développement de la production agricole et agro-alimentaire de l'île,
- Promouvoir une agriculture compétitive dans une dynamique agro-écologique,
- Sauvegarder la biodiversité dans les écosystèmes naturels.

En quelques chiffres : le Cirad à La Réunion possède deux sites principaux, à Saint-Denis et à Saint-Pierre. Il possède 35 hectares de terrains d'expérimentations sur quatre stations. Plus de 280 personnes y travaillent, dont 170 agents permanents (50 chercheurs et 120 techniciens).

3. L'équipe ARTISTS

Le stage s'est déroulé au sein de l'équipe ARTISTS, qui fait partie du **département Persyst**. Ce département conduit des études sur les productions tropicales. Ses travaux sont réalisés en partenariat avec des acteurs locaux de la recherche en Afrique, Asie, Amérique Latine et dans les Départements français d'Outre-Mer.
Au sein de ce département, subdivisé en 12 unités de recherche, l'équipe fait partie de l'**UPR Aïda** (Unité Propre de Recherche : Agro-écologie et Intensification Durable des cultures Annuelles).

L'UPR Aïda se positionne sur l'intensification et la durabilité de la production des cultures annuelles en milieu tropical contraint. Ses recherches visent la pleine valorisation des ressources disponibles, en mobilisant les processus écologiques qui régissent leur dynamique au sein des agrosystèmes. Cette UPR se subdivise en cinq équipes de recherches, dont l'équipe ARTISTS.

L'**équipe ARTISTS** (Télédétection, Systèmes d'Information, Techniques de Simulations et Analyses Spatiales) est animée par Pierre Todoroff, chercheur en modélisation et télédétection. Elle comprend : Agnès Tendero, ingénieure en informatique et responsable des projets SI et d'outils d'aide à la décision, Mickaël Mézino, ingénieur en informatique et administrateur systèmes et bases de données, Lionel Le Mézo, ingénieur en télédétection, Louis Paulin, technicien et éco-physiologiste, Raymond Nativel, technicien et météorologiste et Dany Derveilher, technicien et machiniste/télédétecteur. L'équipe a pour ambition de

faire sauter les verrous du regroupement de données spatialisées, de diverses natures et origines, dans les conditions d'accès limité à la ressource numérique des Pays du Sud. Elle s'attache à mettre à disposition ces données pour les producteurs, les analyser, mettre en évidence les lois du fonctionnement des agroécosystèmes et traduire celles-ci en modèles et outils d'aide à la décision qui mettent en œuvre ces données, à l'usage des agriculteurs et décideurs pour améliorer la productivité des agrosystèmes. Deux axes structurent ses activités :

- La gestion de l'information spatiale
- Les modèles et outils d'aide à la décision

La pénurie de données et la difficulté d'accès dans les Pays du Sud ont motivé l'équipe à développer des solutions originales, à la fois méthodologiques et technologiques. L'équipe avait déjà mis en place le portail *Margouill@*, permettant le stockage, la maintenance et la consultation/extraction de données pour les utilisateurs. Grâce à la plateforme AWARE, il est maintenant possible pour les chercheurs d'offrir une bonne visibilité à leurs travaux de recherche. Ce catalogue est ouvert au grand public, mais il est néanmoins possible de restreindre l'accès à certaines ressources si besoin.

Ce catalogue est donc au cœur des motivations qui animent l'équipe. Les missions proposées durant ce stage vont également dans ce sens : ouvrir l'accès aux données, tout en respectant le désir de confidentialité ressenti par certains chercheurs.

III. Le sujet du stage

1. Présentation de l'outil AWARE

a. Pourquoi AWARE ?

Cet outil a été développé pour plusieurs raisons. Premièrement, au Cirad les équipes de recherche utilisent des données géographiques, mais l'accès à certaines données reste compliqué. Avant la création d'AWARE, il était nécessaire de faire une demande à l'équipe ARTISTS ou à la DSI pour y accéder. En y ayant accès, il arrivait parfois que les métadonnées ne soient pas correctement saisies, rendant l'utilisation délicate pour des chercheurs. D'autre part, avant de faire une demande, il fallait également avoir conscience de l'existence de ces données. En créant un tel catalogue, les chercheurs peuvent voir rapidement toutes les données existantes, et peuvent faire une recherche (Fig.3) par titre (A), par type (B), par catégorie (C), par mot-clé (D), par propriétaire (E), par date (F), par région (G), ou encore par emprise spatiale (H). Il devient très facile de trouver la donnée qu'ils recherchent, mais aussi de découvrir des données qui sont susceptibles de les intéresser. Enfin, la création de ce catalogue est également un moyen pour le chercheur de faire connaître les résultats de son travail. Son travail en est valorisé, et est stocké sur un serveur et non sur son pc.

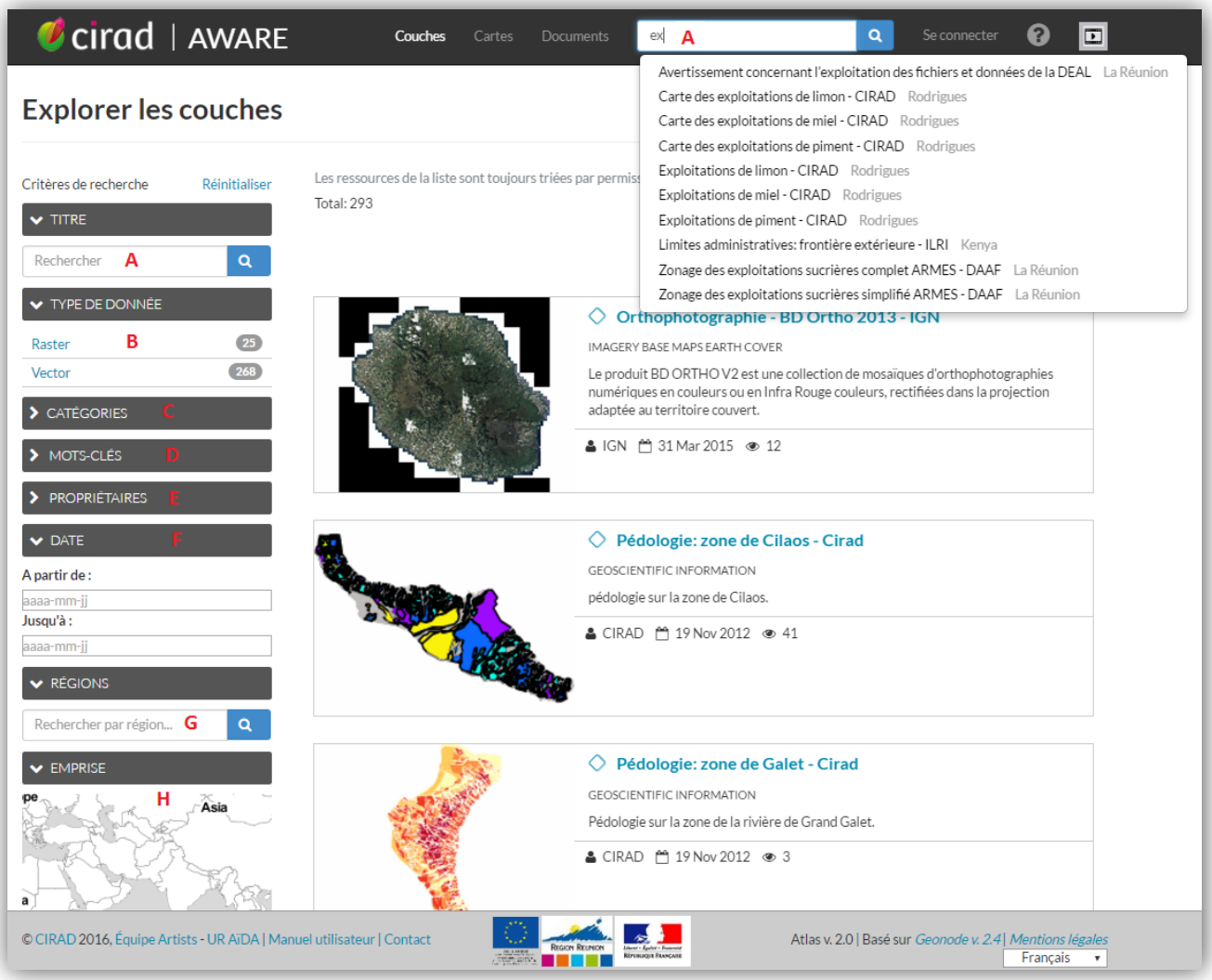


Fig. 3 - Catalogue de couches d'AWARE, et filtres proposés

La force d'AWARE réside dans sa capacité à mutualiser l'information spatiale du CIRAD, grâce aux contributeurs. Ces derniers peuvent ajouter des couches et créer des cartes sans utiliser un logiciel SIG. Il

est possible de lier des documents (expertises, rapports, photos, etc.) aux couches et cartes produites. Enfin, l'atlas utilise des solutions technologiques libres qui implémentent les standards OGC[1] (WMS, WCS, WFS et OWS) garantissant l'interopérabilité avec des logiciels SIG comme QGIS ou ArcGIS[2]. Il est tout à fait possible de charger des couches du catalogue AWARE depuis QGIS par exemple, et les permissions seront propagées dans le logiciel : seules les couches autorisées seront proposées à l'utilisateur.

Il est également possible de moissonner d'autres catalogues, afin de proposer un contenu plus riche. L'atlas suivant la directive INSPIRE[3], il peut aussi être moissonné par d'autres serveurs cartographiques.

b. Situation à l'arrivée

La plateforme AWARE est basée sur l'IDS[4] open source *GeoNode (http://geonode.org/)*. Une IDS est une plateforme dédiée à la gestion et à la publication de données spatialisées. Après avoir étudié plusieurs IDS, l'équipe a choisi celle-ci car elle désirait mettre en place un atlas cartographique web basé sur des technologies libres et sans coût de licence. Ce catalogue doit en effet pouvoir être transférable aux acteurs des Pays du Sud avec lesquels l'équipe travaille, dont certains ne peuvent pas forcément assumer une solution sous licence privée. Le but de l'équipe est bien de faciliter l'accès à la donnée pour ces acteurs. GeoNode a ce double intérêt d'avoir une licence libre, et d'être personnalisable. La contrepartie est le temps de travail passé à installer, à comprendre puis à développer cet outil. L'interface est relativement simple d'utilisation, ce qui permet aux utilisateurs non-spécialistes de partager leurs données spatialisées et de créer des cartes facilement. Les outils de gestion intégrés à GeoNode permettent la création de données, de métadonnées, et la visualisation de ressources.

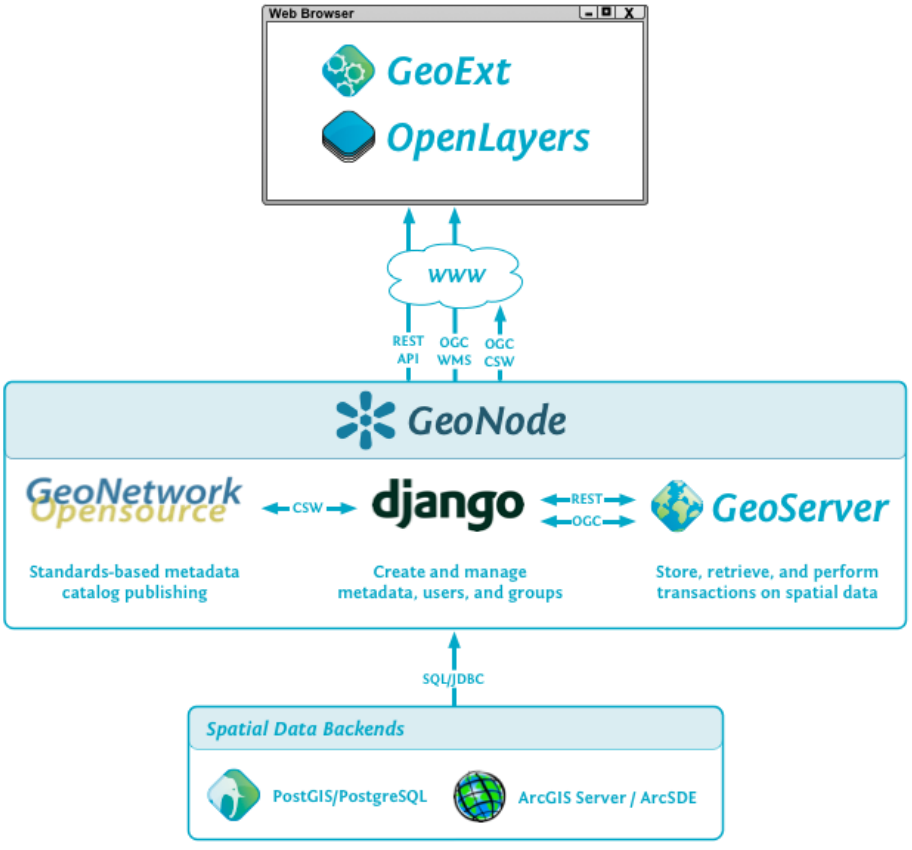


Fig. 4 - Architecture simplifiée de GeoNode (autre schéma plus détaillé en Annexe). Source : https://geonode.readthedocs.io/en/1.2/developers/architecture.html

[1] Open Geospatial Consortium : Consortium international pour développer et promouvoir des standards ouverts, afin de garantir l'interopérabilité des contenus, des services et des échanges dans les domaines de la géomatique et de l'information géographique. Un glossaire plus détaillé est proposé à la fin de ce rapport.
[2] Voir http://aware.cirad.fr/help/.
[3] Directive visant à favoriser l'échange des données au sein de la Communauté européenne.
[4] Infrastructure de Données Spatiales

Par rapport à l'instance originale de GeoNode, un certain nombre de spécificités avaient déjà été développées quand je suis arrivé en stage, par Hugo Ferrari, qui a mis en place le catalogue à partir d'avril 2015 dans le cadre d'un stage du M2 SIGMA.
Les principales modifications sont d'un côté une grande ouverture du catalogue, puisque toutes les ressources y sont visibles, pour n'importe quel utilisateur, connecté ou non. Attention, cela ne veut pas dire pour autant que n'importe qui a le droit de consulter toutes les ressources ou de les télécharger, mais tout le monde peut voir que la donnée existe dans le catalogue. Pour n'importe quelle ressource, un utilisateur a le droit de télécharger les métadonnées, même s'il n'a pas le droit de consulter la ressource. De cette manière, on permet à une personne de savoir si une donnée est susceptible de l'intéresser : elle a accès au titre, au *thumbnail*[5] et aux métadonnées (résumé, date de publication, description des attributs pour les couches, etc.).

Dans GeoNode, le comportement original est plus contraignant : soit on ouvre tout le catalogue, et tout le monde peut voir toutes les ressources du catalogue, mais également les consulter, soit au contraire les personnes qui n'ont pas le droit de consulter certaines ressources ne verront même pas qu'elles existent, elles seront invisibles dans le catalogue. L'équipe ARTISTS souhaitait un comportement moins tranché. Ainsi, toutes les ressources sont affichées dans le catalogue. En plus de cela, un tri en fonction des ressources a été implémenté. L'utilisateur a face à lui une liste de ressources triées spécifiquement pour lui, avec celles qu'il peut consulter en premier, puis celles à accès restreint ensuite.

Une autre évolution par rapport à GeoNode concerne les actions des utilisateurs. Sur GeoNode, les utilisateurs peuvent contribuer à leur guise, à partir du moment où ils sont connectés. Sur la version originale, j'ai essayé de créer un groupe d'utilisateurs « non-contributeurs », auquel j'ai donné comme unique droit celui de voir les ressources, mais cela ne change rien, quiconque est connecté a accès à ces boutons :

Fig. 5 – Actions autorisée sur une instance de GeoNode originale

Hugo Ferrari avait donc fait en sorte que seuls les utilisateurs ayant la permission « add_resourcebase » (ajouter des ressources) puissent avoir accès à ces boutons. S'ils sont non connectés ou non contributeurs, ils auront à la place un bouton « Explorer des ressources ». Ces boutons permettant de contribuer apparaissaient sur la page d'accueil du site, et sur les catalogues. Mais en cliquant sur une ressource, d'autres boutons existent : un utilisateur simplement connecté peut « Créer une carte » (Fig.6) à partir d'une couche. En faisant cela, il peut ajouter des cartes dans le catalogue en enregistrant son travail, alors même qu'il n'est pas contributeur. Hugo avait également modifié ce comportement, autorisant seuls les contributeurs à accéder à cette fonctionnalité, et affichant un message bloquant pour tout autre utilisateur cliquant sur le bouton. En étant contributeur, il faut en plus avoir le droit d' « Éditer la ressource » pour accéder à la fonctionnalité.

[5] Image miniature de la ressource

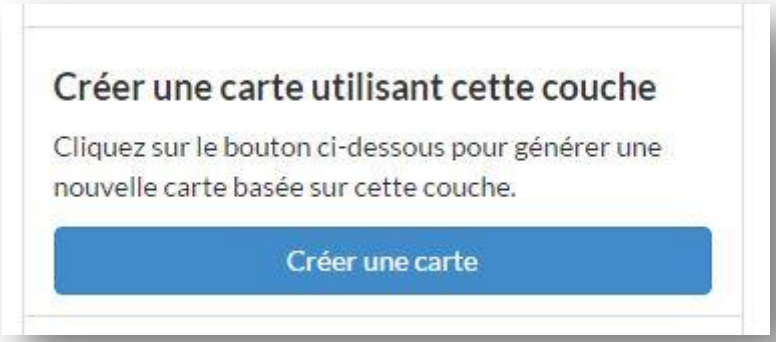


Fig. 6 - Bouton permettant de créer une carte à partir d'une ressource. Sur AWARE, il est visible lors de la consultation d'une couche mais uniquement pour les contributeurs

L'équipe a également modifié le code pour proposer une vraie recherche par catalogue. Sur GeoNode, il y a quatre zones pour effectuer une recherche par texte : la barre de recherche présente en haut de page dans la barre de navigation (visible sur toutes les pages), puis les barres latérales présentes dans le catalogue des couches, des cartes et des documents (Fig.3).
Ce sont ces barres réparties dans les trois différents catalogues qui nous intéressent ici. Elles proposent les mêmes résultats dans la liste déroulante de la barre de recherche, ce qui est plutôt incohérent : si on recherche le document « Manuel d'utilisateur » dans la barre principale, le résultat apparaîtra dans la liste déroulante car cette barre effectue une recherche sur tous les catalogues. Si l'on se positionne maintenant dans le catalogue des couches et que l'on recherche la même chose, le « Manuel d'utilisateur » apparaîtra dans les propositions de titres de la liste déroulante, ce qui est trompeur puisqu'en validant avec la touche <Entrée>, aucun résultat ne sera en réalité retourné dans le catalogue.
Après avoir modifié le code, la recherche au sein des catalogues n'affiche plus que les ressources du catalogue dans lequel on se trouve dans la liste déroulante. Ainsi, les titres qui apparaissent correspondent tous à ceux présents dans le catalogue des couches par exemple, si l'on se trouve dans ce catalogue. Le « Manuel d'utilisateur » ne sera ainsi plus proposé.
L'équipe en a également profité pour permettre à l'utilisateur d'accéder directement à la ressource en cliquant sur un titre qui apparaît dans la liste déroulante de la barre de recherche.

Un export des métadonnées aux formats .html et .txt a également été ajouté. GeoNode propose un téléchargement des métadonnées au format XML selon différents standards : ISO, FGDC, ebRIM, Dublin Core, DIF et Atom – conçus pour être « lus » par des machines. Le téléchargement aux formats .html et .txt permet d'avoir en un clic les métadonnées dans un format préalablement mis en page proprement, et surtout lisible tel quel.

Concernant le catalogue, il comprenait 289 couches, 17 cartes et 44 documents à mon arrivée.

2. Objectifs du stage

L'objectif principal de ce stage est de développer de nouvelles fonctionnalités sur la plateforme AWARE, d'en améliorer la confidentialité, et d'évaluer le coût (temps de travail, régressions possibles) d'une migration vers la dernière version stable de GeoNode - 2.4 - sortie en novembre 2015. L'équipe ARTISTS souhaiterait également effectuer une remontée de sa version à la communauté GeoNode, une fois toutes ces évolutions achevées, en essayant de faire en sorte que cette dernière intègre un maximum de fonctionnalités d'AWARE pour la prochaine version 2.6. Cela leur permettrait d'éviter tout un travail de remise à niveau s'ils décident de passer à la version 2.6 plus tard. Si cette remontée n'est pas possible, ou si la communauté n'est pas intéressée, il faudrait néanmoins rendre cette version accessible sur internet, d'une part pour que quelqu'un puisse la télécharger et l'installer s'il est intéressé (du Cirad, ou extérieur au Cirad), et d'autre part pour respecter le contrat de licence qui impose de remettre à disposition de la communauté les modifications sur le code source d'un projet open source. Enfin, l'autre objectif de ce stage est de compléter la traduction de la plateforme.

A mon arrivée, un certain nombre d'évolutions à développer a préalablement été listé. Certaines sont mineures, d'autres sont vraiment importantes car elles modifient le fonctionnement de la plateforme. Il est ainsi possible de dresser deux listes :

Evolutions majeures :

- **Masquer les boutons si l'utilisateur n'a pas les permissions requises** : à mon arrivée, il est possible pour n'importe quel utilisateur de cliquer sur le bouton « Créer une carte » par exemple, même s'il n'en a pas la permission. S'il a la permission, il est redirigé vers la page de création de carte. S'il ne l'a pas, un message bloquant lui informe qu'il n'a pas les droits. Plutôt que d'afficher ce message bloquant, il est préférable de directement masquer le bouton.
- **Ajouter une section « Derniers ajouts » sur la page d'accueil** : pour rendre le site plus dynamique, on souhaite afficher les 5 ou 10 dernières contributions (de couches seulement).
- **Récupérer l'adresse mail de l'utilisateur lors de sa première connexion** : s'il fait partie de l'annuaire *LDAP*[6] et que le champ « e-mail » n'est pas rempli sur son profil AWARE, son adresse mail sera automatiquement enregistrée sur son profil à partir de celle présente dans l'annuaire LDAP. Cela permet de faciliter par la suite l'envoi de mails de l'administrateur aux utilisateurs.
- **Améliorer la visibilité sur le site**, en permettant de ne pas faire apparaître la ressource lorsque l'on vient de la publier : on souhaite proposer une solution pour que le contributeur ait le choix de publier ou non la ressource. Il peut ainsi en différer la publication jusqu'à ce que les métadonnées soient complétées.
- **Résoudre un bug présent sur la version d'AWARE** en production à mon arrivée en stage : un utilisateur d'AWARE s'ajoute automatiquement, indépendamment de sa volonté et de la nôtre, lorsque l'on définit les permissions sur les ressources. Si l'on ne fait pas attention et que l'on clique sur enregistrer les modifications, cet utilisateur obtient tous les droits sur ces ressources.
- **Compléter la traduction du site** : comprendre le fonctionnement de la traduction avec GeoNode, et y contribuer.
- **Effectuer la migration de la plateforme vers la version 2.4 stable de GeoNode**, après avoir fait une étude de coût/bénéfices d'une telle migration.
- **Mettre en place l'import de fichiers *.sld***[7], permettant d'avoir automatiquement les bons styles lorsque l'on ajoute une couche sur la plateforme.
- **Passer le site en *SSL***[8] (https) : en l'état, la plateforme n'offre pas encore de transferts sécurisés. Les informations échangées transitent donc en clair sur le réseau.
- **Permettre à l'administrateur d'ajouter des ressources et de modifier les permissions** sans qu'il ait besoin de s'intégrer lui-même dans un groupe qui a toutes les permissions : il s'agit d'une anomalie présente sur le site en production durant mon stage.

Evolutions secondaires :

- **Agrandir le cadre de visualisation des ressources** : il s'agirait de reprendre le format qui existait dans d'anciennes versions de GeoNode, plus confortable pour la navigation sur les couches et les cartes.
- **Ajouter un bouton « Zoomer sur l'emprise de la couche » sur ce cadre** : il existe en l'état un bouton « Zoomer sur la carte max », qui permet de zoomer sur l'emprise du planisphère, ce qui n'est pas vraiment utile et satisfaisant.
- **Faire apparaître la légende** lors de la visualisation d'une couche, sans avoir à cliquer sur le bouton « Légende » présent sur ce même cadre.
- **Rajouter les logos des partenaires en bas de page d'accueil** : ils sont présents dans le *footer* (barre fixe présente en bas de page du site), mais très peu lisibles. Il faudra les garder, mais également les rajouter de manière à ce qu'ils soient lisibles en bas de page d'accueil, en dehors du footer.

[6] Lightweight Directory Access Protocol
[7] Styled Layer Descriptor
[8] Secure Sockets Layer

- **Mettre en place un meilleur contrôle sur la saisie des mots-clés** : il arrive que les mots-clés soient mal interprétés. Exemple : « Occupation », « du », « sol » au lieu de « Occupation du sol ».
- **Faire défiler le carrousel présent sur la page d'accueil automatiquement**, et non manuellement.
- **Rendre la saisie de la « Région » obligatoire lors de la complétion des métadonnées**, par le biais d'un message bloquant dans le formulaire.
- **Permettre le *scroll*[9] dans la liste déroulante de la barre de recherche**.
- **Changer l'affichage du nom des utilisateurs** : afficher leur nom au lieu de leur pseudo, pour plus de confidentialité et de sécurité sur la plateforme.

Pour organiser mon temps de travail et les tâches à effectuer au cours de ce projet, nous avons dès le début du stage créé un diagramme de Gantt.

[9] Déroulé vertical

Fig. 7 - Diagramme de Gantt

Ce diagramme a évolué au fil du projet, pour plusieurs raisons. Tout d'abord parce que la migration vers la version 2.4 stable de GeoNode a été faite dès le début du stage, avec Mickaël Mézino. Au lieu d'évaluer le coût/bénéfice d'une telle migration, nous l'avons installée directement car elle comptait un nombre

important d'améliorations, et parce que certaines tâches qui m'avaient été assignées allaient être résolues par le simple fait d'effectuer la migration, les rendant ainsi inutiles, et nous faisant gagner du temps pour la suite du projet. Il sera également plus simple de remonter notre version à la communauté si on est au même niveau d'avancement qu'elle.
L'autre raison pour laquelle le diagramme a évolué au cours du projet, c'est justement parce que la remontée à la communauté a dû se faire de manière anticipée. Les développeurs de GeoNode ont « gelé » la branche Master mi-juillet afin de se concentrer sur le développement de leur version 2.6, prévue pour être mise à disposition fin août/début septembre.

D'une manière plus générale, pourquoi passer à la version 2.4 stable ? Il sera plus simple de passer de la version 2.4 stable à la version 2.6 stable plutôt que de la version 2.4 *pre-release* à la version 2.6 stable. Cette migration inclut le passage à GeoServer 2.7, avec des apports importants, comme l'API REST d'ArcGIS maintenant supportée, ou encore un meilleur support du SLD - même si cela reste encore incomplet. Enfin, certains problèmes sont corrigés dans cette version (création des thumbnails revue, enregistrement des mots-clés en partie résolu, on pensait également que la gestion des permissions irait dans le sens de celle de notre instance, mais finalement non).

Le programme à mon arrivée était le suivant :

- Prise en main des outils et de la plateforme aware.cirad.fr,
- Développement de nouvelles fonctionnalités,
- Passage ou non à la version 2.4 stable,
  Si oui :
  - Réintégration de nos spécificités,
  - Réintégration de notre catalogue,
- Clonage du site en développement,
- Remontée à la communauté GeoNode de nos spécificités,
- Mise en production du site en développement.

Pour finalement aboutir à celui-ci :

- Prise en main des outils et de la plateforme aware.cirad.fr,
- Passage à la version 2.4 stable,
- Réintégration de nos spécificités,
- Réintégration de notre catalogue,
- Développement de nouvelles fonctionnalités (1),
- Clonage du site en développement,
- Remontée à la communauté GeoNode de nos spécificités,
- Développement de nouvelles fonctionnalités (2),
- Mise en production du site en développement.

3. Environnement et méthodes de travail

a. Environnement

Pour réaliser ce projet, j'évolue à la fois sur un système Windows (7), et Linux Ubuntu Server (14.04). Que ce soit le site en production ou le site en développement, ils sont localisés sur un serveur virtuel, accessible par le protocole sécurisé *SSH*[10]. J'utilise le client SSH pour Windows *Putty (http://www.putty.org/)* pour y accéder. Je pourrais utiliser l'éditeur de texte *Vim*, via le terminal Linux, pour modifier les fichiers du projet, mais je préfère utiliser *Notepad++* sur Windows, pour sa simplicité d'utilisation.

[10] Secure SHell

Les modifications du code source se font donc via *Notepad++* (*https://notepad-plus-plus.org/download/v6.9.2.html*), qui grâce à son plugin « NppFTP » permet d'enregistrer les modifications en direct, et évite d'avoir recours à un logiciel comme *FileZilla* par exemple. Il possède en plus de cela un grand nombre de plugins, notamment un dont je me sers presque quotidiennement : « Compare », qui permet d'afficher les différences entre les lignes de code de deux fichiers.

GeoNode est basé sur *Django* (https://www.djangoproject.com/), un framework open source de développement web en Python. Django est spécialement dédié à la création de site web, et rend celle-ci plus simple. La philosophie de ce Framework, c'est le « DRY » (Don't Repeat Yourself) : on part du principe que les développeurs ont à peu près tous les mêmes besoins pour la création de sites, et que la plupart font les mêmes erreurs. Dans ce cas, autant utiliser un Framework qui a préalablement résolu ces erreurs et qui rend les choses beaucoup plus faciles grâce à sa bonne organisation. Django a une architecture dite « MVC » (Modèle – Vue – Contrôleur), parfois appelé (pour ce framework) « MVT » (Modèle – Vue – Template).
En plus de Django, GeoNode a d'autres dépendances, nécessaires pour la communication avec les serveurs spatiaux. Le catalogue affiché dans GeoNode est le reflet de celui de GeoServer. GeoNode est là pour rendre plus simple la gestion, le catalogage, la création de cartes, la recherche sur le catalogue de données, et la création de métadonnées. Les données spatiales ainsi que les services OGC sont implémentés et gérés par GeoServer. Grâce à Django et aux bibliothèques Python, GeoNode est continuellement aligné sur le catalogue GeoServer, et les échanges concernant les permissions et la sécurité sont instantanés.

Concernant le serveur, AWARE est accessible et utilisé par le public durant mon stage. Je ne peux donc pas le modifier sans risquer de pénaliser les utilisateurs. Un clone de ce serveur a donc été mis à ma disposition et me sert de serveur de développement. Ce clone m'a surtout servi avant la migration vers GeoNode 2.4 stable, car une fois effectuée, la nouvelle version de développement sur laquelle je travaillais était une toute nouvelle instance de GeoNode, et n'était plus du tout en phase avec le serveur de production.
Le serveur de production est virtualisé sur l'hyperviseur *ESXI VMware*, dont l'accès est strictement réservé à l'administrateur, tandis que je travaille sur *VirtualBox*, un hyperviseur grand public, clone de la machine de production.

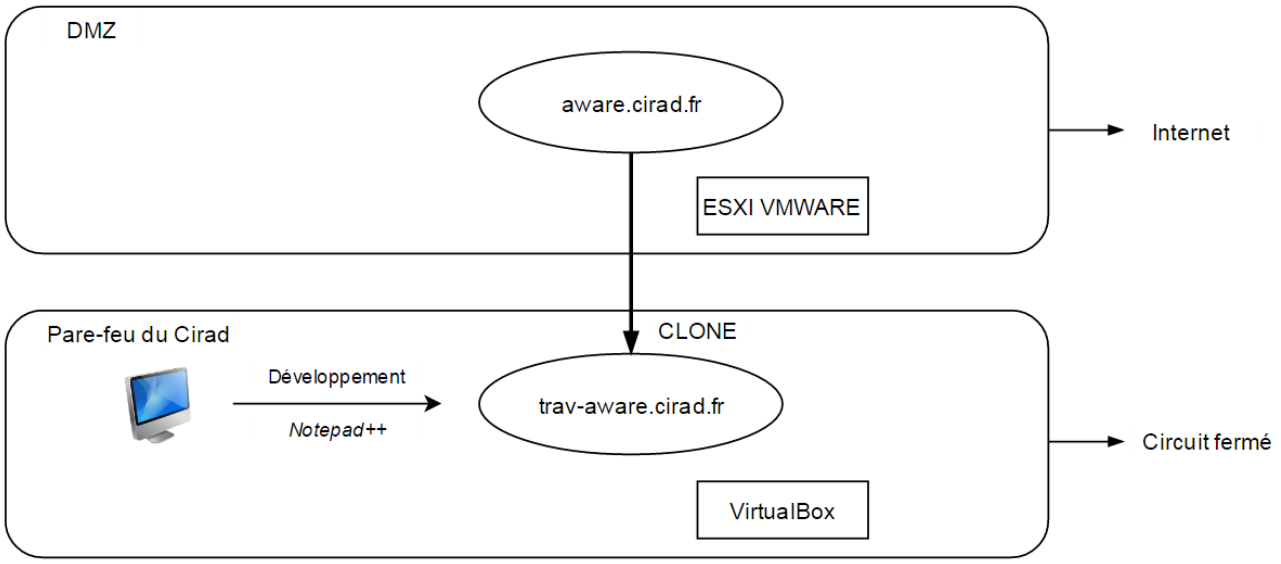


Fig. 8 - Diagramme expliquant l'organisation pour le développement du site aware.cirad.fr

b. Méthodes de travail

Pour définir les tâches qui me sont attribuées, l'équipe ARTISTS a recours à *Mantis Bug Tracker*. C'est un outil de suivi des anomalies détectées durant les phases de test/recette d'une application. Il permet de répertorier des points à améliorer, décrire des évolutions et cela sous forme de fiches. Chaque fiche est assignée à quelqu'un, qui peut commenter, puis signaler à l'équipe (Agnès Tendero, Mickaël Mézino et moi-même) lorsqu'une étape est effectuée, ou que la tâche est terminée. Ces fiches précisent la priorité de la tâche (de faible à élevée), et son impact (mineur à majeur, voire critique). C'est un outil très pratique permettant d'avoir une bonne visibilité sur l'avancement du projet. Il permet enfin d'avoir tout l'historique des échanges, et les étapes à réaliser pas à pas pour développer une fonctionnalité/résoudre une anomalie. Nous verrons cependant que toutes les fiches ne sont pas commentées – parfois elles n'expliquent pas comment procéder pour résoudre un problème. De plus, il arrive que certaines évolutions soient développées sans qu'une fiche soit créée.

0000309	3		Evolutions	**majeur**	résolu (rolland)	2016-08-18	Mécanisme de modération
0000324	1		Anomalies	mineur	affecté (rolland)	2016-08-16	Régressions : création d'une carte
0000318	1		Anomalies	mineur	fermé (rolland)	2016-08-16	Régressions : page d'accueil
0000325	1		Anomalies	mineur	fermé (rolland)	2016-08-16	Régression : mail à l'administrateur
0000329			Evolutions	mineur	affecté (rolland)	2016-08-12	Derniers ajouts
0000326			Anomalies	mineur	affecté (rolland)	2016-08-12	Régression : autorisations
0000315	1		Evolutions	mineur	fermé (rolland)	2016-08-04	Zone de recherche globale
0000299	1		Evolutions	mineur	fermé (rolland)	2016-08-04	LDAP : récupération de l'email

Fig. 9 - Affichage de fiches sur Mantis Bug Tracker

Le projet suit plusieurs étapes clés. La plus longue et laborieuse est le **développement**. Sur la durée du stage, à peu près deux mois ont vraiment été consacrés à cette étape, mais en réalité le développement s'est fait de manière continue tout au long du stage, parfois en arrière-plan (lors de la remontée de notre version à la communauté notamment). Certaines fonctionnalités sont plus simples que d'autres à développer, et à évaluer en termes de temps de travail. D'autres fonctionnalités en revanche prennent plus de temps que prévu, car elles nécessitent de se plonger dans les forums de développeurs ou dans la documentation des différents langages utilisés. Une fois tous les développements effectués, on peut passer en phase de **tests**. Agnès, Mickaël et moi-même réalisons des tests à chaque nouvelle évolution apportée. En revanche, la phase de test finale est effectuée sans moi, afin d'avoir un regard objectif sur le site. Cette étape a duré un peu moins d'une semaine. Durant cette étape, un certain nombre de fiches sont créées sur Mantis Bug Tracker et me sont assignées. Je peux ensuite entamer la phase de **correction**. Cette dernière phase m'a pris à peu près une semaine également. Une fois les corrections testées pour vérifier les non régressions, on réalise une dernière sauvegarde du serveur, pour ensuite le mettre en production.

Pour le site en production et celui en développement, nous utilisons le même nom de domaine (http://aware.cirad.fr). Si nous avions utilisé un nom de domaine différent pour la version de développement, ce nom aurait été enregistré et propagé dans les liens avec la base de données. Lors de la mise en production, il y aurait eu incohérence, perte, voire impossibilité d'accéder à certaines fonctionnalités.
Or durant la phase de développement, il est essentiel de pouvoir passer de la plateforme AWARE en développement à celle en production, afin de pouvoir comparer l'avant/après, et ainsi pouvoir s'assurer qu'une modification sur la version en développement n'entraîne pas de régressions par rapport à celle en production. Pour cela, l'opération consiste à forcer la résolution du nom de domaine localement, en rajoutant la correspondance d'adresse IP avec notre nom de domaine dans le fichier Host de Windows. Pour utiliser le site en ligne, il suffit de mettre en commentaire la ligne rajoutée dans ce fichier. Une autre opération consiste à entrer directement l'adresse IP du site en production dans la barre de recherche du navigateur. Dans ce cas le site est accessible mais comporte quelques erreurs (thumbnails qui ne s'affichent pas par exemple). Enfin, une dernière solution est d'utiliser un bureau distant sur Windows, et dans ce cas il n'est pas nécessaire de modifier le host ou d'entrer l'adresse IP du site en production dans le navigateur.

IV. Réalisation et résultats

1. Le passage à GeoNode 2.4 stable

a. A l'origine : veille technologique

Dans le cahier des charges initial j'étais censé faire de la veille technologique sur la nouvelle version de GeoNode, pour évaluer le coût en temps de travail d'un passage de notre version à la version 2.4 stable. Il fallait également que je m'intéresse aux nouvelles fonctionnalités proposées, et celles abandonnées. Il est probable qu'en passant à une nouvelle version, il y ait des régressions, c'est-à-dire des pertes par rapport à celle actuellement utilisée en production. Ce sont ces pertes qu'il faut quantifier, lister et évaluer. Même s'il n'y a pas de pertes, il faut alors réintroduire toutes les spécificités que l'équipe avait développées dans la nouvelle version, c'est-à-dire repartir d'une nouvelle instance de GeoNode vierge. Finalement, la priorité a été donnée à la migration.

b. Migration vers GeoNode 2.4 stable dès le début du stage

Pourquoi effectuer cette migration dès le début du stage ? Cela implique en effet beaucoup de travail : réintroduire d'emblée toutes les modifications effectuées par mon prédécesseur dans le nouveau code, et réimporter tout le catalogue. Il était en réalité important de passer à cette nouvelle version rapidement, car sans même faire de veille technologique, il apparaissait clairement qu'elle n'avait que des avantages : un nouveau système de permissions annoncé, les groupes et les permissions par groupe ont été repensés, les services distants ont été améliorés, les *templates*[11] sont plus réactifs et ont un nouveau design, de nouveaux filtres sont proposés pour les ressources (par propriétaire par exemple), GeoServer 2.7 vient remplacer la version 2.5, et beaucoup d'autres améliorations qui concernent la performance de GeoNode. Une autre raison est que certaines évolutions prévues seraient devenues inutiles puisque résolues par le simple fait d'effectuer la migration. Cela fait gagner du temps, ces développements n'étant plus à faire.
Techniquement, pour installer la nouvelle version 2.4 stable, il suffit d'aller sur le site de GeoNode, et de suivre les instructions. Il s'agit d'une série de lignes de commande à entrer dans le terminal Linux, et d'effectuer ensuite diverses configurations. Alors que la version qu'avait installée Hugo Ferrari était une version *pre-release*, de développement, celle que nous avons installée cette fois est une version stable. Nous verrons plus tard que si la différence n'a pas eu d'impact pendant un certain temps, elle nous a causé des soucis par la suite.

Effectuer la migration dès le début du stage s'est avéré être un excellent exercice pour moi, car j'ai dû me plonger totalement dans le code de notre version, pour comprendre son organisation et ses spécificités.

2. La réintégration des spécificités liées à la version en production

a. Introduction

Il y a plusieurs manières de réintégrer les modifications effectuées préalablement :

Avec Notepad++, le principe consiste à ouvrir deux fenêtres, et faire une comparaison de fichiers grâce au plugin « Compare » installé. Je n'ai pas d'autre choix que d'utiliser cette méthode, car la documentation laissée par mon prédécesseur est malheureusement assez pauvre comparée au volume de fichiers réellement modifiés. Le principal problème auquel j'ai été confronté est le suivant : quels sont les fichiers qui ont été modifiés, et à quoi sert leur modification ? Dans quel ordre faut-il modifier les fichiers, et quelles sont les dépendances ?

[11] Modèle, gabarit de page web

Lorsqu'on installe GeoNode sur un serveur, tout se fait en ligne de commande, en suivant le script d'installation présent sur le site. Une fois installé, Hugo n'a pas eu recours à un gestionnaire de version.
On ne peut donc pas faire de comparaison du code qui existait à l'installation en avril 2015 (Fig.10), et de celui après toutes les modifications effectuées. Si cela avait été le cas, il aurait été relativement simple de ne rajouter que les éléments qui ont changé. Le problème est qu'Hugo avait installé une version développeur de la 2.4, qui n'est plus accessible étant donné que la branche Master en constante évolution. En faisant une comparaison du code de la version en production et de la version 2.4 stable (présente sur GitHub), il est très difficile de savoir s'il s'agit de modifications effectuées par l'équipe, ou bien de modifications effectuées par la communauté GeoNode (et il y en a tous les jours).

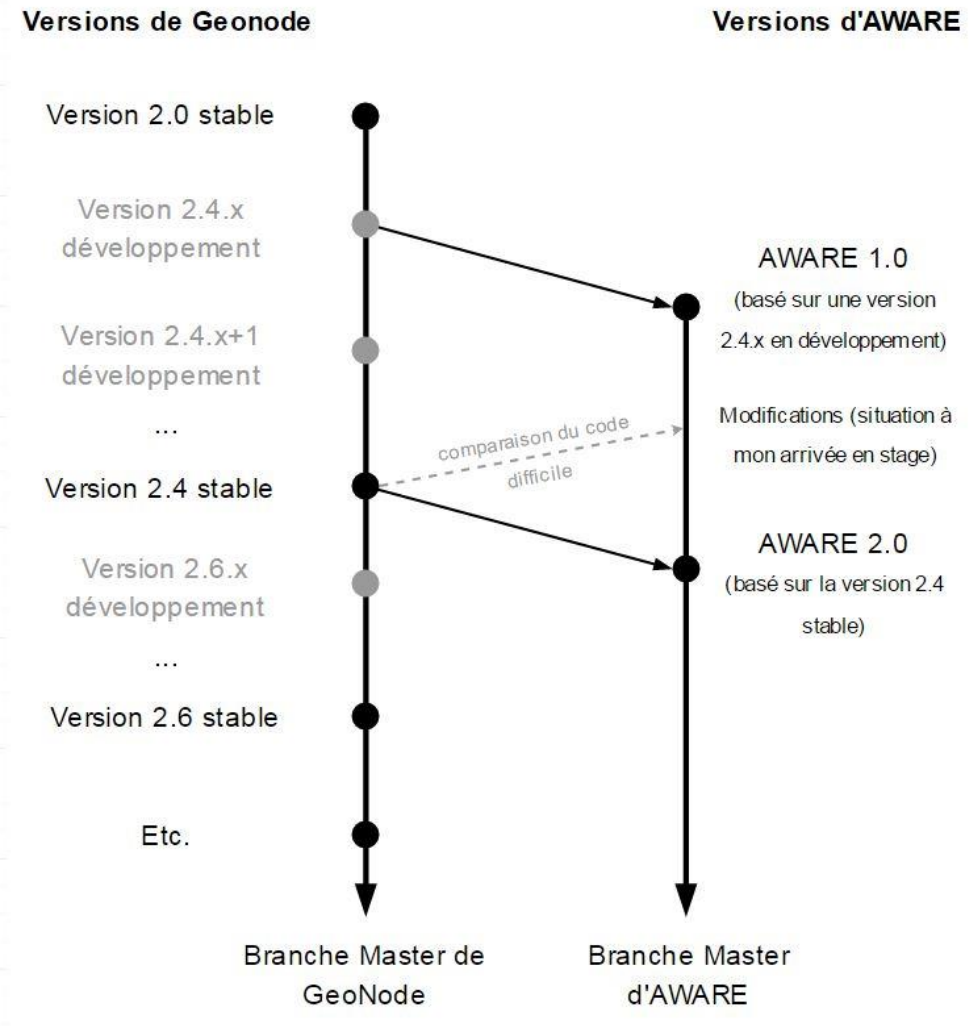


Fig. 10 – Versions de GeoNode, et versions d'AWARE

Les fonctionnalités à réintégrer sont celles que nous avons énoncées précédemment, lorsque nous évoquions les spécificités d'AWARE par rapport l'instance originale de GeoNode. Certaines modifications ne concernent pas des spécificités de notre instance à proprement parler, mais elles demandent un réel travail de développement, car elles ne sont effectives que suite à l'installation de dépendances, qui sortent du cadre de GeoNode. Je pense notamment à la connexion à la plateforme via l'annuaire LDAP, que j'expliquerai plus tard dans ce rapport. En effet, ce type de connexion est prévu par GeoNode, mais il est nécessaire d'avoir au préalable un client mail installé sur le serveur, ce qui n'est pas forcément précisé sur la documentation de GeoNode.

Afin de bien m'adapter au code, j'ai d'abord commencé par étudier et réintégrer les spécificités plus « esthétiques » et « cosmétiques » que fonctionnelles, c'est-à-dire plutôt les fichiers .css et .html. Une fois

que j'ai mieux maîtrisé l'organisation des fichiers au sein du projet, et mieux compris leur utilité, j'ai commencé à me pencher sur les fichiers .py (Python) et .js (JavaScript).

b. Méthodes pour la réintégration

Il a été assez déroutant de procéder à la réintégration des modifications effectuées avant mon arrivée. Certaines fonctionnalités nécessitent d'intervenir sur plusieurs dizaines de fichiers, situés dans des répertoires différents, et dont un certain nombre de fichiers .py peuvent faire planter la plateforme à la moindre mauvaise indentation (entre autres).

Méthode 1 – Via Mantis Bug Tracker

La première méthode consiste à utiliser l'historique des modifications laissé par l'équipe et visible via Mantis Bug Tracker. Il contient les tâches attribuées à Hugo Ferrari lors de son stage, et la manière dont il les a résolues, théoriquement du moins. C'est ce que j'ai fait dans un premier temps, mais cette méthode a vite montré ses limites : seules certaines modifications ont été répertoriées. Quand elles ont été répertoriées, elles n'ont pas forcément été commentées : la fiche est marquée comme « résolue », mais la partie « commentaire » est restée vide, car il n'est pas obligatoire de la remplir.

Méthode 2 – Via des recherches effectuées sur le terminal Linux

Les modifications sont traçables directement dans le code, à chaque fois qu'Hugo a modifié celui-ci, un commentaire précède la modification, avec les initiales « HF ». Ainsi, pour retrouver ces modifications, il suffit de se placer à la racine du projet, et de taper la commande :

```
$ rgrep « HF »
```

Cette commande permet d'afficher tous les fichiers contenant les caractères entre les guillemets (en respectant la casse). J'ai pu réintégrer un bon nombre de modifications de cette manière. Cependant, cette méthode montre aussi ses limites rapidement. Beaucoup de modifications n'ont pas été commentées, et quand elles ont été commentées, il n'y a pas toujours d'explications. Enfin, le code « HF » est assez court, et les résultats retournent bien souvent des fichiers compilés ou minifiés[12] dans lesquelles peuvent apparaître ces deux lettres. Finalement, en tapant cette commande, il faut aller chercher les quelques endroits où cela concerne effectivement une modification qui a été faite.

Méthode 3 – Via une comparaison de code

La troisième solution a été de faire un listing de tous les fichiers modifiés par mon prédécesseur, quelques jours après l'installation de GeoNode (car le jour même de l'installation, à peu près 2200 fichiers sont modifiés du fait de leur création).

```
$ find geonode –mtime -350 >
/home/utilisateur/modifications_effectuees.txt
```

Cette commande m'a permis de dresser une liste d'une centaine de fichiers qui ont été modifiés depuis le début des modifications d'Hugo. L'opération consistait ensuite à utiliser le plugin « Compare » pour rechercher les lignes de codes différentes entre les deux versions.

Finalement, trois semaines de travail auront été nécessaires pour tout réintégrer, avec l'aide de Mickaël Mézino, et en ayant recours à ces trois méthodes. La réintégration de nos spécificités a été une des tâches les plus compliquées à réaliser. Mais d'un autre point de vue, cela a été la façon la plus efficace de rentrer dans le code, et de le comprendre, même si c'était de manière assez pénible, cela m'a beaucoup fait progresser. J'ai rapidement pris conscience que commenter son code est essentiel, pour soi comme pour les autres, et qu'il est assez facile d'oublier de le faire.
Pour un futur projet, je pense que c'est une bonne chose d'utiliser un logiciel de gestion de version (tel que Git par exemple), qui permet d'avoir une bonne traçabilité.

[12] Un fichier minifié est un fichier dans lequel tous les espaces et les commentaires ont été supprimés. Cela permet de l'alléger grandement, ce qui est utile pour certains fichiers qui contiennent un très grand nombre de caractères.

Autre idées pour gérer de futures modifications dans un projet : j'ai réfléchi à une manière de dresser une liste claire et précise des modifications effectuées dans un projet, très facilement traçable. Un code bien documenté est extrêmement important : s'il n'est pas documenté, il peut devenir presque impossible à réutiliser par quelqu'un d'autre. S'il est bien documenté, cela lui assure une certaine pérennité. Il suffit d'écrire, pour *chaque modification*, un code comme celui-ci :

```
<!-- AR**2401** -->
```

Où :

AR = Initiales de la personne qui a apporté une modification
24 = Indicatif qui précise que c'est une modification (nombre sans importance, ici choisi au hasard)
01 = Identifiant unique de la modification (plus le code est précis, plus les résultats retournés le seront).

Si l'on effectue un `rgrep "AR**24"`, on obtiendra une liste de toutes les modifications effectuées par Antoine Rolland, sans aucun résultat trouvé dans des fichiers minifiés ou compilés car le code est suffisamment précis. Si l'on effectue un `rgrep "AR**2408**"`, on obtiendra une liste de tous les fichiers qui ont été modifiés pour le développement de la fonctionnalité « 08 ». Par exemple, pour développer la création d'une section « Derniers ajouts » sur la page d'accueil, on pourrait rajouter le code `<!-- AR**2408** Ajout des lignes suivantes pour creer une section 'derniers ajouts' sur la page d accueil -->` (en HTML) dans chaque ligne ou groupe de lignes ajoutées ou modifiées dans un ou plusieurs fichiers. De cette manière, la traçabilité sera très bonne, et ce n'est pas vraiment contraignant. Il suffit ensuite de se faire un petit répertoire avec les modifications que tout le monde a effectuées, et on retrouve tout très rapidement.

3. Réintégration du catalogue

a. Réintégration du catalogue en lignes de commandes

Nous pensions au début réintégrer tout le catalogue (289 couches, 17 cartes et 44 documents) en lignes de commandes, c'est-à-dire prendre toutes les couches présentes sur la version d'AWARE 1.0, pour les intégrer dans la nouvelle version basée sur GeoNode 2.4 stable. Lorsque nous installons la nouvelle version, le catalogue est vide, cela implique la création d'une nouvelle base de données. Finalement, nous n'avons pas trouvé de moyen satisfaisant de le faire en cherchant dans la documentation de GeoNode. D'autre part, nous craignions que cela ne modifie des liens dans les bases de données, qui sont assez sensibles. De plus, GeoNode prend encore mal en compte les fichiers .sld, nous nous attendions à ce que les styles soient très mal importés. Nous avons donc opté pour une réintégration manuelle du catalogue, que nous avions évaluée à plus ou moins deux semaines de travail. Etant donné que nous avons pris cette décision rapidement, nous n'avons pas perdu de temps puisqu'il aurait de toute manière fallu passer un certain temps pour trouver une solution viable et sécurisante pour réimporter tout le catalogue.

b. Réintégration manuelle du catalogue

J'ai donc procédé à la réintégration manuelle. Cela a été l'occasion de corriger des erreurs qui étaient parfois présentes sur certaines couches (sur les couches elles-mêmes ou bien leurs métadonnées). Cette tâche m'a finalement pris deux semaines, ce que nous avions prévu initialement.

Pour réintégrer le catalogue, j'ai téléchargé les couches directement sur le site en production, et les ai uploadés sur le site en développement. Pour ce qui est des couches Raster, cela se fait d'une manière différente : il faut passer directement par GeoServer et indiquer le chemin vers l'emplacement du fichier. Ces fichiers étant beaucoup plus lourds (parfois 50 Go), le serveur n'accepte pas leur transfert car ils est supérieur à 1 minute de chargement.

Les couches Raster engendrent des erreurs lorsque l'on se rend ensuite sur le site. En effet, leur import créée automatiquement un thumbnail dans un dossier spécialement dédié mais dont nous ne sommes pas les propriétaires (le propriétaire est le serveur Apache utilisé par GeoNode). Pour cette raison, il est nécessaire de changer les droits d'écriture afin de résoudre ces problèmes : passer de www-data:www-data à www-data:notre_groupe.

Cette partie du projet n'était pas la plus complexe à proprement parler. Cependant, elle m'a pris un certain temps. J'ai constaté quelques disfonctionnements sur certaines couches vecteurs, et certains rasters (erreurs Django/Python, GeoServer, noms de couches invalides, etc.). La manipulation des données reste assez sensible, mais tout a finalement bien été réintégré.

4. Développement et évolutions de l'atlas cartographique AWARE

Le développement de nouvelles fonctionnalités sur AWARE est l'objectif principal de mon stage. Nous diviserons cette partie en quatre sous-parties distinctes, afin de différencier les types d'évolutions. Les premières sont des évolutions qui ont été mises en place par le simple fait d'avoir procédé à la migration vers GeoNode 2.4 stable. Nous verrons ensuite les évolutions que j'ai développées moi-même, avec tout d'abord les principales, qui modifient plus le comportement d'AWARE, puis les secondaires, qui sont plus des évolutions sur l'ergonomie et le design de la plateforme. Enfin, une dernière sous-partie sera consacrée à la traduction d'AWARE, qui nous le verrons peut se faire de différentes manières.

a. Les améliorations suite à la migration vers GeoNode 2.4 stable

Faire apparaître la légende

Nous voulions faire apparaître la légende lors de la visualisation d'une ressource. Sur la version en production, la légende était visible en cliquant sur le bouton « Légende » sur les outils présents sur le cadre de visualisation de la ressource. C'était assez peu intuitif, et la légende s'affichait dans une nouvelle fenêtre. Finalement, l'apparition de la légende « en dur » était prévue dans la version 2.4 stable. Nous n'avons donc pas eu de développement spécifique à réaliser.

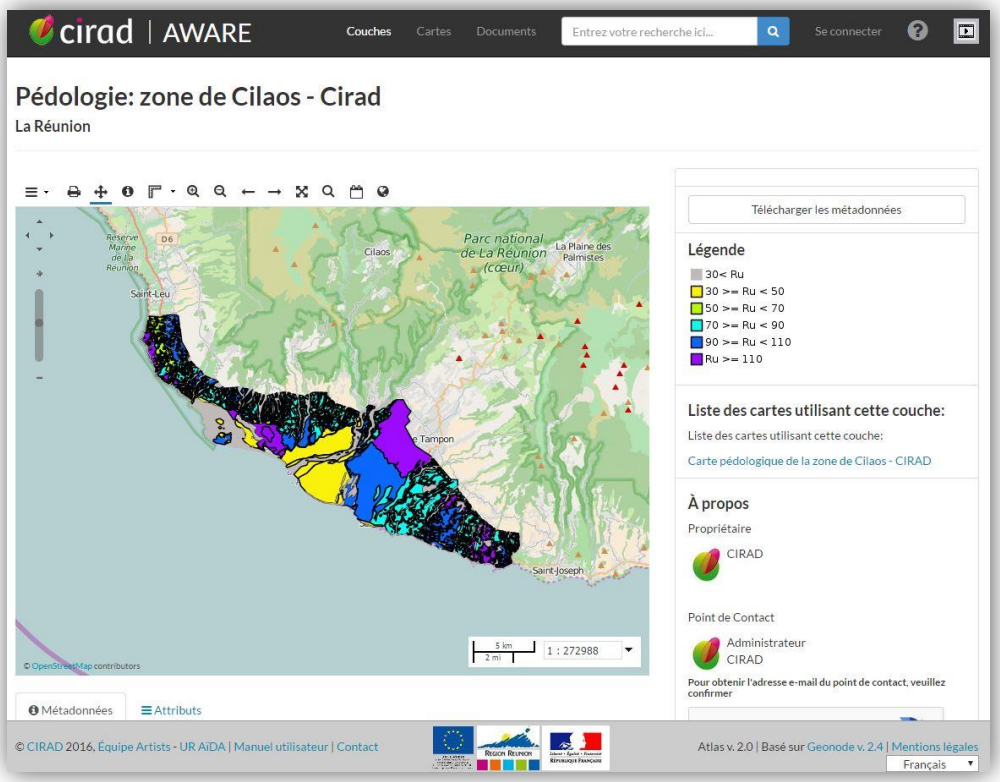


Fig. 11 – La légende est maintenant affichée automatiquement lors de la visualisation d'une couche

Gestion des droits (correction d'un bug présent)

Sur la version en production, un bug apparaissait lors de la définition des droits sur une ressource. En effet, un des utilisateurs d'AWARE se plaçait automatiquement - en dehors de sa volonté et de la nôtre - lors de la définition des permissions pour les ressources. Si l'on ne faisait pas attention et que l'on validait, cette personne obtenait effectivement tous les droits pour ces ressources. Ce bug a été résolu automatiquement en passant à la version 2.4 stable.

Passage à GeoServer 2.7

Le fait de passer à GeoServer 2.7 a augmenté de manière significative les performances du site, dont la vitesse de chargement des ressources, améliorant l'expérience de l'utilisateur lors de sa navigation sur les couches du catalogue. Toutefois, il faudra vérifier si ce n'est pas dû au fait que sur la version de développement nous sommes en réseau local, sur une boucle réseau qui ne quitte pas le bâtiment. En passant en production, nous passerons cette fois par le réseau internet. Il y a tout de même des chances pour que cette rapidité se retrouve plus tard en production, car au début de mon stage, je travaillais sur le clone de la version en production, et les deux affichaient les mêmes performances, alors que l'un était sur le réseau local, et l'autre sur le réseau internet. Enfin, la différence peut également être due au nombre de cœurs alloués à la machine virtuelle.
En passant à GeoServer 2.7, un certain nombre de disfonctionnements qui étaient à résoudre ont été corrigés de fait. La prise en compte des fichiers de style .sld a été améliorée, même si ce support n'est toujours pas complet.

b. Les évolutions principales et/ou fonctionnelles

Zoom sur l'emprise de la couche

Il existe dans GeoNode un bouton « Zoomer sur la carte max », dont l'utilité est assez discutable. Il permet de zoomer sur l'étendue maximale de la carte, c'est-à-dire… l'étendue du planisphère. Ainsi, il est assez déroutant lorsque l'on se trouve à l'échelle précise d'une ville, de cliquer dessus et de se retrouver avec une emprise contenant tous les continents. La seule utilité serait si l'on travaille sur une couche dont l'emprise correspond à celle du planisphère, mais même dans ce cas, un « zoom sur l'emprise de la couche » donnerait le même résultat. De ce fait, j'ai ajouté un bouton « Zoomer sur l'emprise de la couche ». Cette modification m'a pris du temps, car il m'a fallu trouver dans quel fichier la faire. Il s'agit finalement du fichier GeoExplorer.js, fichier extrêmement dense présent dans les fichiers « Static » de GeoNode, en dehors du répertoire principal situé dans Django.

Récupération de l'e-mail d'un utilisateur du Cirad

Chaque agent du Cirad est répertorié dans un annuaire LDAP : un protocole de service d'annuaire permettant le partage de bases de données sur un réseau. Ces bases de données contiennent des informations sur les personnels, organisations et matériels à gérer. Ils permettent notamment l'authentification et la réplication[13] lorsqu'on se trouve sur internet.
Le LDAP comprend également des personnes extérieures au Cirad.

Pour s'authentifier sur AWARE, il existe deux méthodes (Fig.12). La première méthode : l'administrateur ajoute un nouvel utilisateur, lui attribue un nom d'utilisateur et une adresse mail. Il l'intègre ensuite dans un groupe (fonctionnement sur lequel repose GeoNode), qui déterminera ses droits (groupe contributeurs, groupe visiteur, etc). Les groupes ne sont pas prédéfinis, c'est l'administrateur qui les créée et qui spécifie leurs droits.

L'autre méthode consiste à utiliser un annuaire LDAP. Pour cela, il est nécessaire d'activer ce service dans le projet GeoNode. Le Cirad utilisant ce type d'annuaire pour référencer ses agents, mais aussi des

[13] Grâce au LDAP, il est possible de se connecter à plusieurs plateformes avec les mêmes identifiants et les mêmes mots de passe, sans même avoir à s'inscrire une première fois : la plateforme reconnaît le couple utilisateur-mot de passe.

partenaires, toutes les personnes y figurant peuvent se connecter à la plateforme. Il suffit de saisir ses identifiants LDAP. En revanche, il n'est pas prévu qu'un utilisateur puisse créer son propre compte.

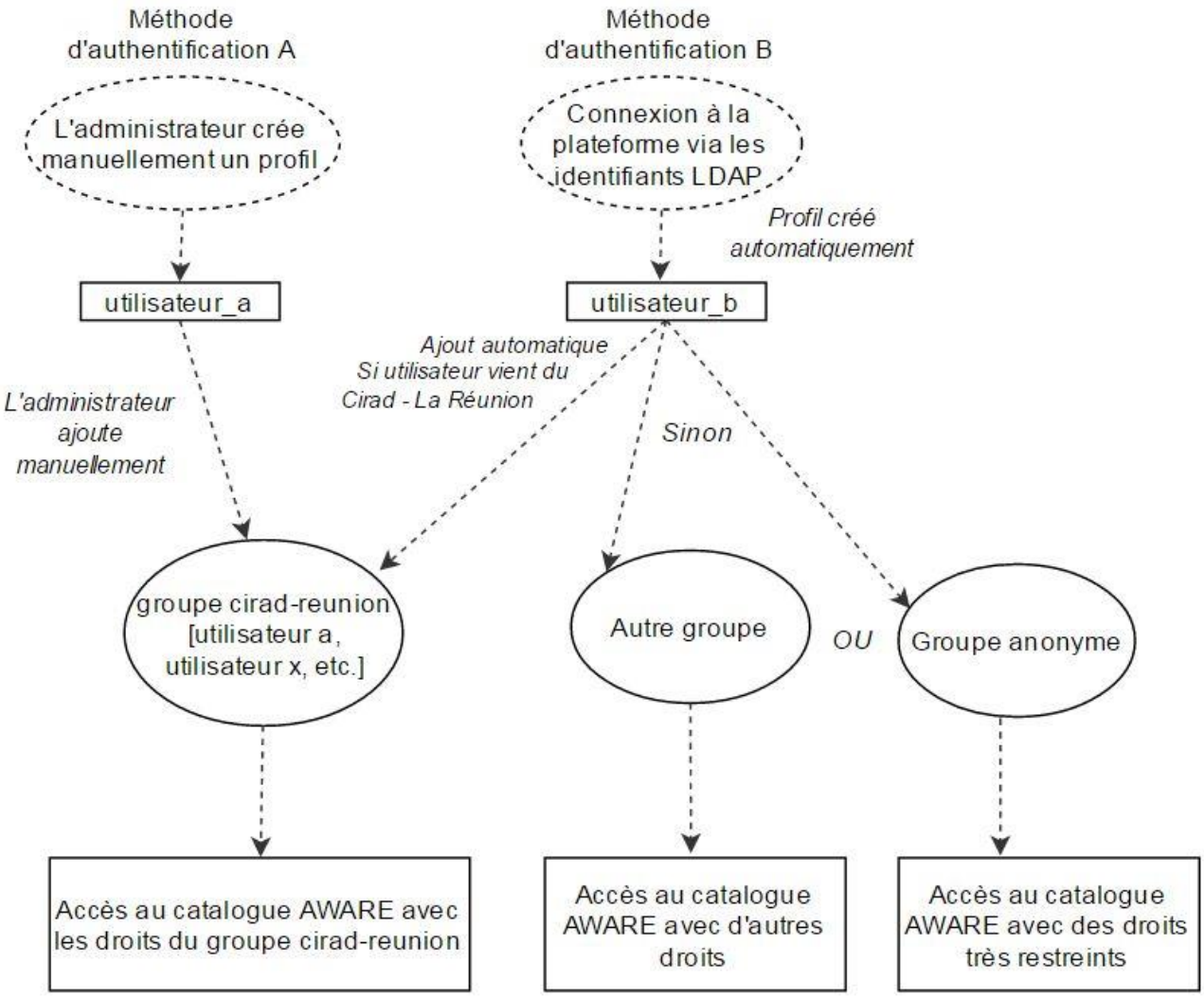


Fig. 12 – Deux scénarios existent pour la connexion à la plateforme AWARE

Hugo Ferrari avait créé une fonction qui permettait, à la première connexion sur le site, de placer automatiquement l'utilisateur dans le groupe « Cirad Réunion » d'AWARE si le champ *organisation* du LDAP comprenait « CIRAD » et le champ *région* comprenait « La Réunion ». Ainsi, à la première connexion, l'utilisateur a déjà les autorisations préalablement définies sans que l'administrateur n'ait à s'en occuper.

Dans le même esprit, j'ai ainsi créé une fonction permettant, dès la première connexion d'un utilisateur référencé dans le LDAP, de récupérer l'adresse mail de celui-ci, puis de l'enregistrer automatiquement dans son profil AWARE dans le champ « e-mail ». J'en ai également profité pour automatiser la récupération du prénom et nom de la personne, ainsi que l'organisation pour laquelle elle travaille, me rappelant à quel point l'encodage peut être un sujet sensible avec le langage Python.

Nous avions besoin de cette fonctionnalité, car lorsqu'un utilisateur utilise la fonction « demander le téléchargement » d'une ressource via AWARE, seul son nom d'utilisateur est transmis dans sa demande. Il est donc impossible de lui répondre puisque l'adresse mail n'est pas renseignée. Maintenant, nous avons automatiquement les adresses mail de chaque utilisateur étant référencé dans l'annuaire LDAP.

Bloquer l'accès à certains boutons qui nécessitent les bonnes permissions

Sur le site en production, l'accès à certains boutons était autorisé, alors que l'utilisateur n'avait pas forcément la permission d'effectuer certaines actions. Par exemple, un utilisateur non contributeur pouvait cliquer sur le bouton « Créer une carte » alors qu'il n'en avait pas le droit, et un message « Vous n'êtes pas

sur la liste des contributeurs à l'atlas » apparaissait. Il était donc préférable de masquer directement les boutons plutôt que d'autoriser le clic et de bloquer l'utilisateur par la suite.
Pour cela, j'ai utilisé une fonction qu'Hugo Ferrari avait créée : « user_can_add_resourcebase ». Elle permet de déclarer qu'un utilisateur a le droit d'ajouter une ressource. Grâce à Django, il est ensuite très simple d'afficher/masquer un bouton en fonction des permissions, en ajoutant quelques lignes :

```
{% load base_tags %} #Chargement du fichier contenant la fonction en question
{% user_can_add_resource_base as add_tag %}
    {% if add_tag %}     #Condition : si l'utilisateur est un contributeur
        <a href="{% url "layer_upload" %}" class="btn btn-primary pull-
right">{% trans "Upload Layers" %}</a> #Affichage du bouton "Ajouter des couches"
(Traduit par GeoNode via Transifex)
    {% endif %}      #Fin de la condition
```

L'ajout de cette condition a dû se faire dans 16 fichiers, répartis dans les sections *couches*, *cartes*, *documents*, *page d'accueil* et *recherche* du projet.

En dehors de cela, le bouton « Ajouter au panier» est apparu avec la migration vers GeoNode 2.4 stable. Il permet de faire une sélection de plusieurs ressources, afin de définir les permissions de manière groupée, ou bien de créer une carte avec plusieurs ressources sélectionnées. Cette nouvelle fonctionnalité a quelque peu perturbé le fonctionnement de notre site, car il rendait possible pour n'importe quel utilisateur de modifier les permissions des couches qui ne lui appartiennent pas. J'ai donc bridé cette fonctionnalité, pour que seul le propriétaire ou la personne ayant les droits de modifier la ressource puisse définir les permissions sur celle-ci, ou créer une carte à partir de celle-ci. Cette fois, il n'est pas possible de le faire via Django, mais via *AngularJS*, un framework Javascript. La condition est définie dans l'attribut « ng-if », propre à AngularJS. On voit ici que le bouton apparaît si : le « cart » (panier) est activé, et l'utilisateur peut modifier les permissions et l'utilisateur peut voir les données de la ressource et la ressource est « publiée ».

```
ng-if="cart && item.can_change_perms && item.can_view_resource &&
item.is_published" #Syntaxe AngularJS, permet d'afficher  le panier sous certaines conditions.
```

Il est primordial de bien paramétrer l'affichage de ce bouton, car sinon la politique de confidentialité et de sécurité de notre plateforme est complètement remise en question. Ainsi, voici l'affichage pour un contributeur ayant tous les droits sur une couche (ici, l'administrateur) :

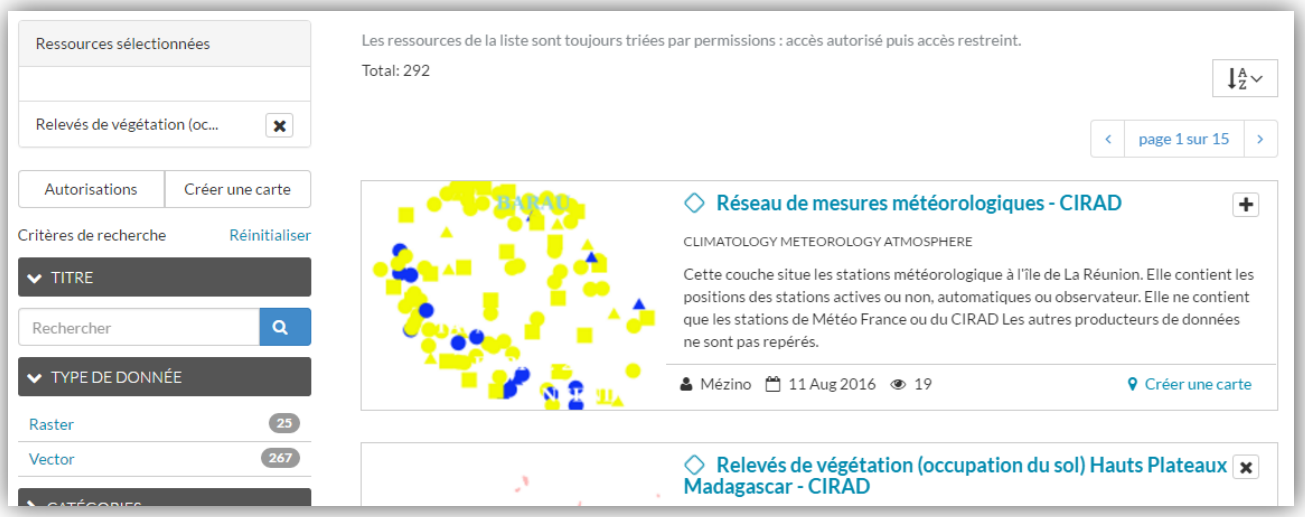


Fig. 13 – L'ajout à la sélection pour un contributeur ayant les permissions nécessaires

Pour un utilisateur non-contributeur, nous avons souhaité rester dans la continuité de ce que nous avions défini ailleurs. Il n'est pas souhaitable pour un utilisateur de voir apparaître un panier dont les boutons sont

inutilisables, je l'ai donc désactivé pour les non-contributeurs, car ils n'ont pas le droit de créer de carte, ni de définir des permissions.

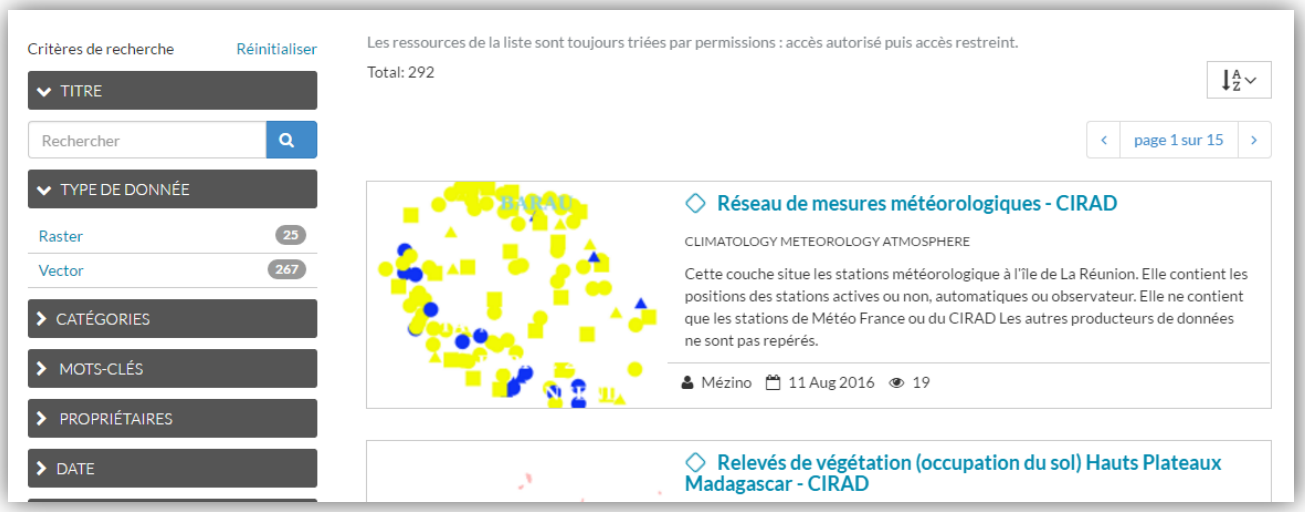


Fig. 14 – L'ajout à la sélection pour un utilisateur non-contributeur (impossible)

Rendre opérationnel le bouton « Est publiée » pour une ressource

L'équipe ARTISTS met l'accent sur la confidentialité et la visibilité des données, aspect très important pour les chercheurs. Certains sont assez réticents à publier le fruit de leur travail sur un site ouvert à tous et voudraient que leurs ressources ne soient pas visibles avant qu'elles ne soient vraiment prêtes à être rendues publiques. Pour cela, j'ai créé un statut spécifique, qui correspond en quelque sorte à une « mise en quarantaine ». Soit le chercheur rend la couche invisible avant de la publier, soit l'administrateur peut décider de rendre une couche invisible car elle ne respecte pas les formats demandés, ou parce qu'elle ne fonctionne pas.

Cette fonctionnalité existait, un bouton « Est publiée » à cocher/décocher est proposé lors de la complétion des métadonnées.

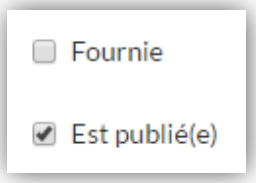


Fig. 15 – Option permettant de rendre une ressource visible ou invisible, proposée dans le formulaire des métadonnées

Mais cocher ou décocher la case était sans effet sur GeoNode. La première tentative pour vérifier le fonctionnement de ce booléen a donc été réalisée sur GeoServer. Si le test unitaire a bien été concluant, cela a également conduit à un effet de bord entraînant un disfonctionnement de GeoNode et empêchant tout ajout de ressource. Nous avons alors dû reprendre une ancienne sauvegarde.

Finalement, j'ai trouvé le moyen de rendre l'activation de ce bouton effective. Ainsi, en comparaison avec l'instance originale de GeoNode, nous avons maintenant trois niveaux de visibilité :

- Ressource invisible sur le catalogue pour un utilisateur lambda. Elle est visible sur le catalogue par le propriétaire de la ressource et l'administrateur. Elle est également visible sur un catalogue alternatif affichant uniquement les ressources en attente ;
- Ressource visible dans le catalogue, mais non consultable en cliquant dessus si l'on n'a pas les autorisations nécessaires. Cela affiche les métadonnées (résumé, date de publication, description des attributs pour les couches, etc.) téléchargeables et le nom de la personne à contacter.

- Ressource visible dans le catalogue et consultable en cliquant dessus, lorsqu'on a les autorisations nécessaires, ou alors si la couche est rendue publique.

J'ai pour cela créé trois nouveaux templates, basés sur ceux des catalogues des couches, des cartes et des documents. L'accès à ces catalogues alternatifs de ressources en attente n'est possible que pour les contributeurs. Si AWARE permet de rendre des couches invisibles, ce n'est pas sans contrepartie : l'équipe ARTISTS ne désire pas héberger un catalogue de couches invisibles, le statut « En attente » a vraiment été créé pour que le contributeur puisse remplir les métadonnées sereinement jusqu'à ce qu'il juge la ressource prête pour la publication. Nous avons donc décidé de désactiver toute autre action sur ces couches : le propriétaire lui-même ne peut pas créer de carte avec par exemple, seulement remplir les métadonnées. Une fois publiée, toutes les actions possibles seront rétablies.

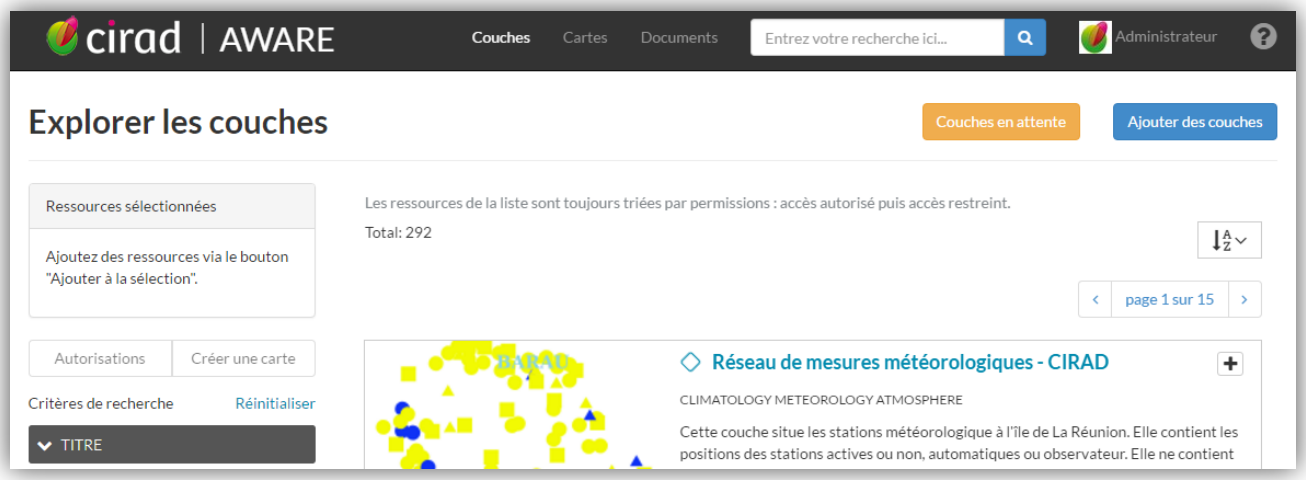


Fig. 16 – Catalogue complet des couches, visible par tous. Le bouton « couches en attente » est seulement visible par l'Administrateur et les contributeurs

Fig. 17 – Catalogue des couches en attente, visible par les contributeurs. L'administrateur y voit toutes les couches en attente, le contributeur y voit seulement ses propres couches en attente

Mettre en avant les « Derniers ajouts » sur la page d'accueil

Cette fonctionnalité a été une des premières que j'ai tenté de développer, et une des dernières que j'ai finalement réussi à faire fonctionner. Durant mon stage j'ai fait de nombreux essais qui s'étaient tous soldés par un échec.
Sur la version 2.0 de GeoNode – que l'équipe n'a jamais installée – il y avait deux sections « dernières couches », et « dernières cartes », qui ont par la suite été supprimées dans la 2.4. Nous voulions réintégrer la section « dernières couches », car elle rend le site plus dynamique, et parce qu'elle génère

automatiquement des activités sur la page d'accueil du site, ce qui est susceptible de le faire remonter sur la page des résultats de recherche de Google.
Vers la fin de mon stage, j'ai décidé de m'intéresser plus sérieusement à la version 2.0. Ayant acquis plus de connaissances sur Django, je suis enfin parvenu à faire fonctionner cette évolution en me basant sur le fonctionnement de la 2.0. Il s'agit en fait de reprendre le template du catalogue de couches, duquel on ne garde que la liste de couches (on supprime filtres, pagination, panier, etc.). L'intégration de cette évolution a nécessité la modification/création de 10 fichiers, HTML, Python et JavaScript. Finalement, la page d'accueil est maintenant celle que l'on voit sur la Figure 19.

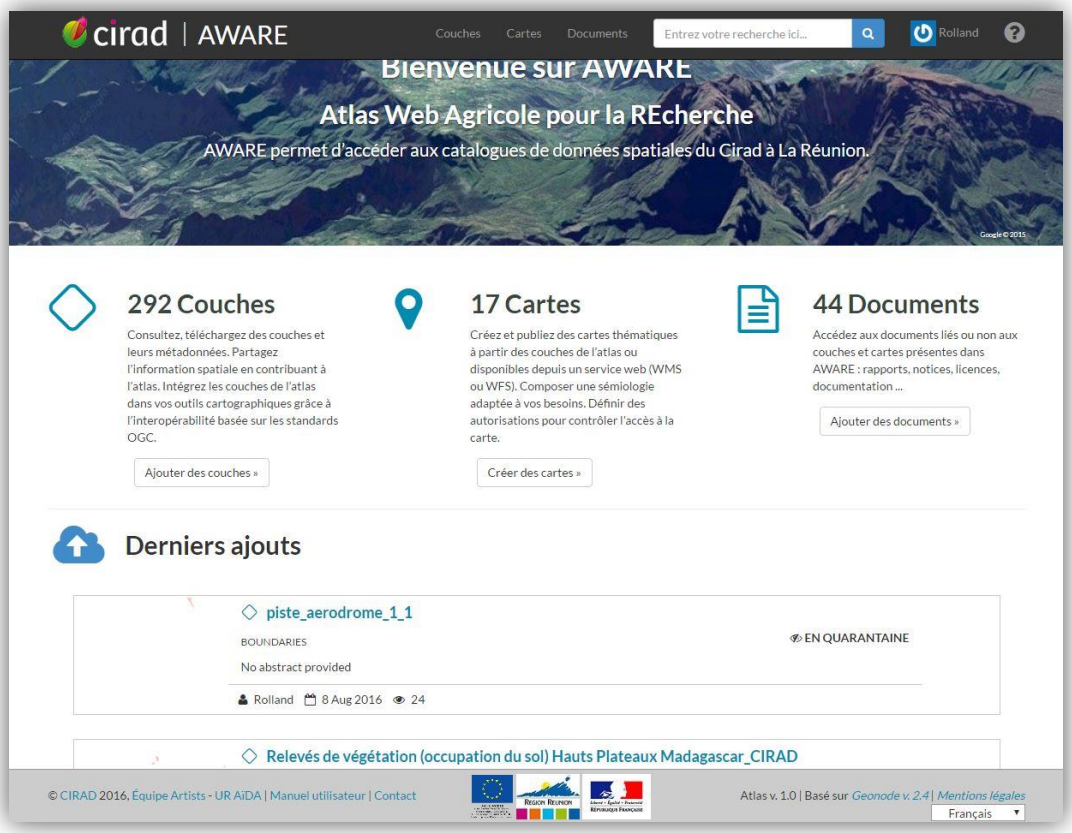


Fig. 18 – Première version de la page d'accueil avec la section « Derniers ajouts »

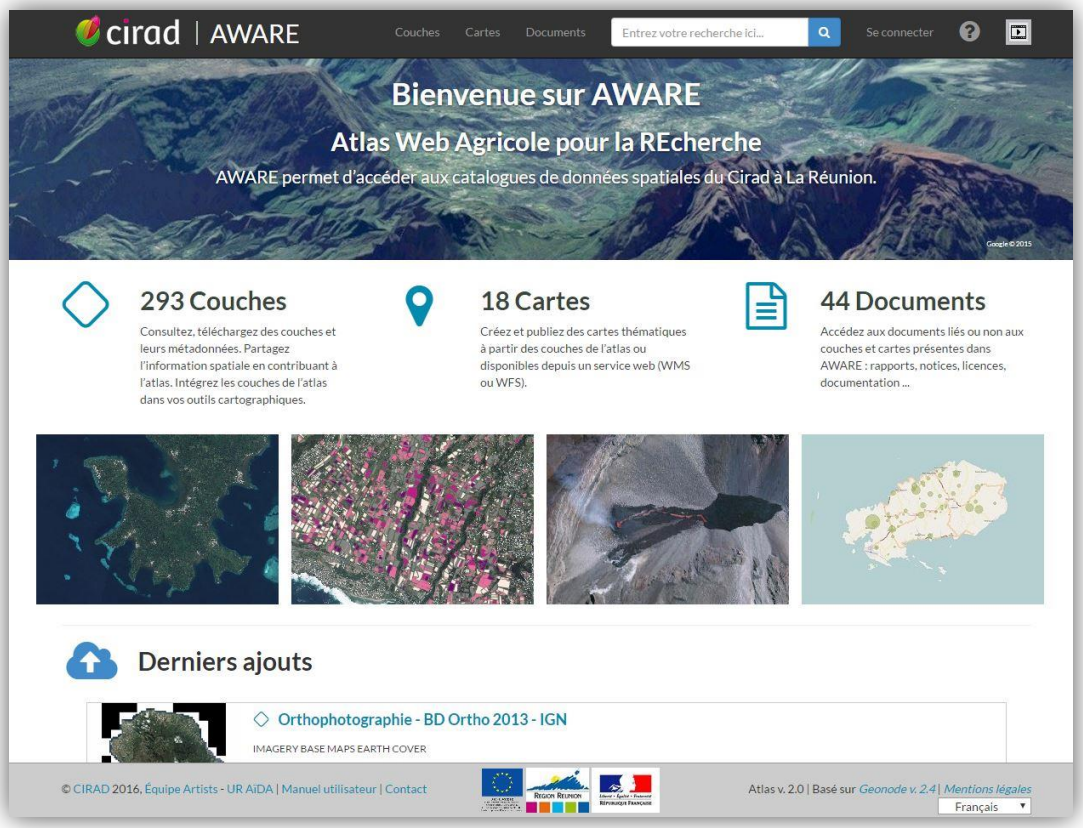


Fig. 19 – Version finalement choisie pour la page d'accueil avec la section « Derniers ajouts ». Les quatre vignettes présentées redirigent l'utilisateur vers des ressources du catalogue lorsqu'il clique

Rendre la saisie de la région obligatoire dans les métadonnées

Dans le catalogue du Cirad, il existe des doublons dans les noms des ressources, car on peut par exemple trouver « Hydrographie : Point d'eau – BDTOPO 2015 IGN » deux fois : pour La Réunion et pour Mayotte. Si l'on ne renseigne pas la région dans les métadonnées, les deux apparaissent dans la liste déroulante de la barre de recherche et il est impossible de les différencier. Pour cette raison, il est essentiel de renseigner la région. Si le champ n'est pas obligatoire, il est possible que certains oublient de la renseigner. A l'origine, seule la « Catégorie » était obligatoire, j'ai donc ajouté la « Région ». Pour ce faire, je me suis inspiré de la Catégorie, ayant remarqué qu'un champ « required » existait pour la Région, je lui ai affecté la valeur « True ». Cette amélioration était donc plutôt simple à réaliser, même s'il faut au préalable passer du temps à trouver dans quel fichier il faut effectuer une modification.

Changement des propriétés de l'Administrateur

Depuis le début du stage, nous ne comprenions pas vraiment pourquoi il était nécessaire – sur notre instance de GeoNode – que l'Administrateur fasse partie d'un groupe. Ce fonctionnement n'est pas naturel : pour pouvoir ajouter des couches et modifier les permissions, il fallait que l'Administrateur soit dans un groupe ayant les permissions, sinon il ne pouvait tout simplement pas effectuer ces actions. Pour cela, nous avions été obligés de créer un groupe « admin » et lui avions attribué toutes les permissions possibles, puis nous nous étions ajoutés nous-même dans ce groupe. Opération assez étrange donc.

En cherchant, j'ai fini par comprendre que cela était dû à deux fonctions qu'Hugo Ferrari avait développées :

- user_can_add_resource_base
- user_can_change_perms

Dans ces deux fonctions, les permissions sont données à des groupes. Etant donné que l'Administrateur ne fait normalement pas partie d'un groupe, il n'aura en conséquence jamais ces permissions. J'ai donc ajouté une condition « si l'utilisateur est l'administrateur » en début de fonction, qui donne automatiquement les permissions à l'administrateur quoi qu'il arrive, et qui lui évite d'avoir à intégrer un groupe.

c. Les évolutions secondaires, et/ou « esthétiques », « cosmétiques »

Agrandissement du cadre pour visualiser les ressources

Le cadre proposé à l'origine par GeoNode est dans un format 16:9ème, très peu confortable lorsque l'on parcoure une couche. Ainsi, j'ai agrandi ce cadre pour revenir à un format s'approchant plus du 4:3, comme GeoNode le proposait dans d'anciennes versions. Étant donné que l'on utilise GeoNode pour visualiser des ressources avant tout, il était important que la navigation se fasse de manière agréable pour l'utilisateur.

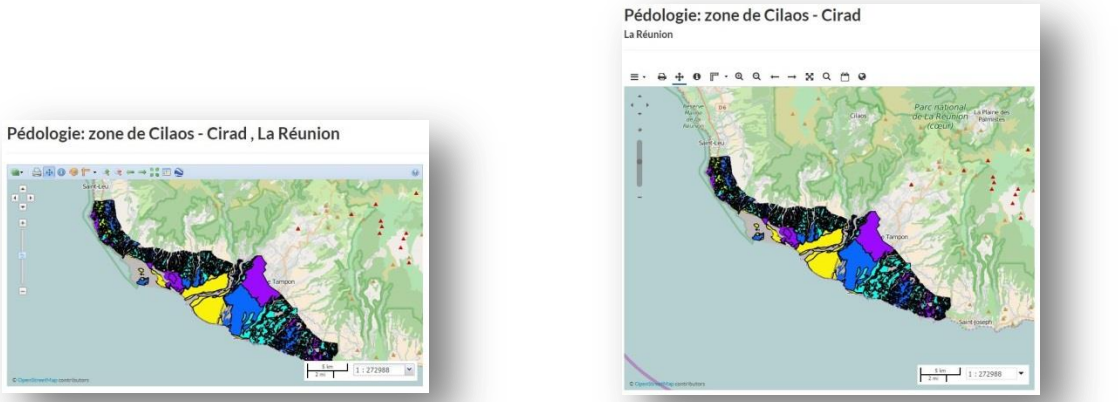


Fig. 20 – Cadre de visualisation : avant et après

Scroll dans la liste déroulante de la barre de recherche

Dans la version 1.0 d'AWARE, le scroll n'était pas possible dans la liste déroulante de la barre de recherche. Si la liste s'affichant dépassait une trentaine de ressources (variable selon la résolution de l'écran du visiteur), on ne pouvait pas voir la fin de la liste, coupée par le bas de la fenêtre du navigateur. J'ai donc modifié les propriétés CSS de la barre de recherche (qui utilise une application tierce : Autocomplete Light) pour intégrer cette fonctionnalité simple, mais efficace et essentielle.

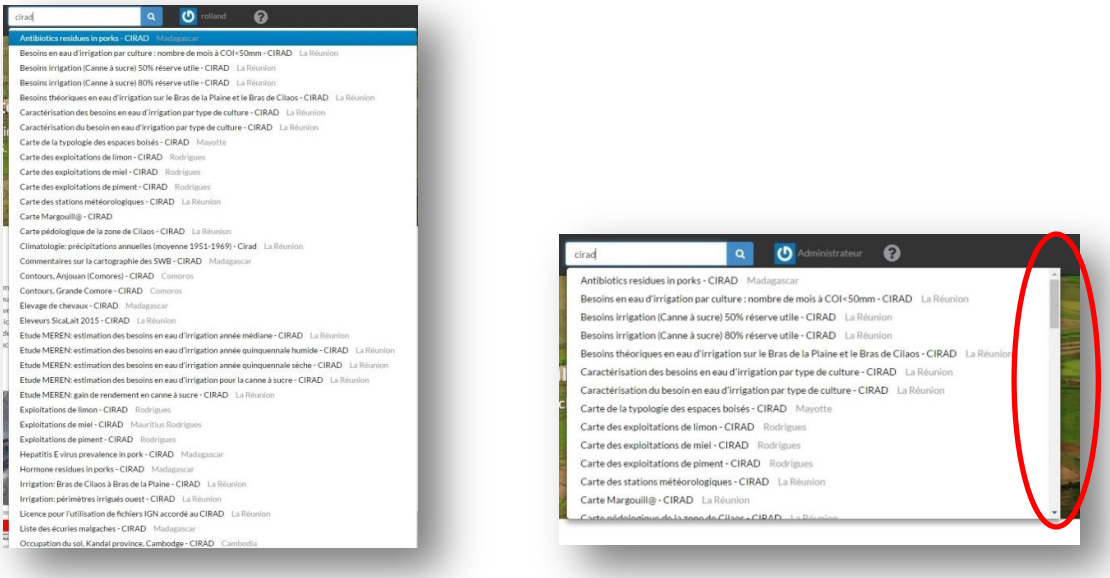


Fig. 21 – Liste déroulante de la barre de recherche : avant et après

Faire défiler le carrousel de la page d'accueil

Le carrousel, bandeau d'images qui défile en page d'accueil, comprend trois photos et du texte expliquant ce que la plateforme AWARE permet de faire. Ces éléments sont censés défiler automatiquement, mais sur le site en production lors du stage, il était figé. L'utilisateur avait cependant la possibilité de cliquer sur les boutons de défilement. En modifiant la propriété CSS du bandeau, j'ai pu réactiver le défilement.

Faire apparaître le <nom> de l'utilisateur plutôt que son <username>

GeoNode affiche le <username> de l'utilisateur à chaque fois qu'il y fait référence. Cependant, en termes de sécurité, ce n'est pas ce qu'il y a de mieux, car il s'agit également de l'identifiant utilisé par l'utilisateur pour se connecter. Un robot qui passerait sur le site aurait donc déjà la moitié du travail de fait s'il veut se connecter via un compte.
De ce fait, j'ai par endroits utilisé le <prenom nom>, et à d'autres endroits le <nom> de l'utilisateur. Cela se fait de différentes manières selon l'endroit où l'on se trouve sur le site : via les tags Django, via des attributs AngularJS, et en modifiant des fichiers Python.

Autres modifications mineures

D'autres petites modifications ont été effectuées au cours du stage, mais ne présentent qu'un intérêt mineur. Dans la version 2.4 stable par exemple, le « Panier » apparaît, ainsi que des boutons « Ajouter au panier », avec des icônes symbolisant un caddie. Etant donné que notre site n'a pas de vocation commerciale (pas d'achat possible directement sur le site), j'ai remplacé ces icônes par de simples « + », et le terme « Panier » par « Sélection ».

Fig. 22 – A gauche l'ajout au panier original, à droite l'ajout à la sélection d'AWARE

J'ai également ajouté un bouton en bas de page permettant de naviguer entre les pages du catalogue. C'est une modification légère, mais qui évite de remonter à chaque fois en haut de page lorsque l'on veut accéder à la page suivante ou précédente.

Enfin, j'ai ajouté un logo de type « checked », qui apparaît lorsque l'on coche un filtre (par exemple le type Vecteur, ou Raster, que l'on peut ensuite combiner à un mot-clé, comme « Occupation du sol »), et qui disparaît lorsqu'on le décoche.

Fig. 23 - Sélection d'un filtre

d. La traduction du site

Tout d'abord, il est important de préciser que sur la version 1.0 d'AWARE, le choix de la langue a été désactivé. Le site s'affiche donc en français, sauf si on y accède depuis une machine dont le système est configuré dans une autre langue. Dans ce cas, seuls les mots traduits dans le cadre du projet GeoNode sont

traduits dans un certain nombre de langues. Les mots et phrases spécifiques à la plateforme AWARE ne sont pas traduits, ils ne sont affichables qu'en français, ce qui est un vrai frein pour les utilisateurs étrangers. Nous verrons que la traduction du site a nécessité le recours à plusieurs méthodes.

Traduction via Transifex

Pour traduire les pages web d'un projet, GeoNode utilise Transifex. C'est un outil permettant « l'internationalisation » d'un site web.

Tout d'abord, il faut se créer un compte sur Transifex, en ligne. Ensuite, rejoindre le projet « GeoNode » présent sur le site de Transifex. On peut alors commencer à contribuer, en choisissant de traduire sur l'interface en ligne, ou de télécharger les fichiers localement pour contribuer via un éditeur de texte.

Lors de l'écriture du code dans un template, il suffit de charger le module « i18n » de Django en début de page :

```
{% load i18n %}
```

Lorsque l'on veut traduire un mot, ou une phrase, on utilise ensuite le code suivant :

```
<a> {% trans "Upload Documents" %} <a>
```

Qui affichera "Ajouter des documents" en français, ou encore « Cargar documentos » en espagnol.

A l'origine, l'idée était donc de compléter le fichier téléchargé comprenant les traductions en français (django.po – fichier Python, qu'il faudra ensuite compiler pour qu'il soit pris en compte). Il existe un répertoire par langue. Une fois complété et compilé, il faut le remonter à la communauté via Transifex, sans quoi les traductions ne sont pas effectives. Le problème est que lorsqu'on le remonte, il faut attendre la sortie de la prochaine version de GeoNode pour que les ajouts soient pris en compte. J'ai donc changé de méthode.

Afin de ne pas avoir à attendre un retour de la communauté, j'ai traduit moi-même les mots que je désirais avoir sur notre site, dans ce fichier django.po. Au début, cela ne fonctionnait pas, car je ne faisais que rajouter des entrées sur le fichier « fr » - français. Les entrées ressemblent à cela.

```
#: templates/index.html:43
msgid "Last contributions"
msgstr "Derniers ajouts"
```

Avec en commentaire le fichier dans lequel le mot est utilisé ainsi que le numéro de la ligne, puis le « msgid » - qui est en fait un identifiant unique du mot, puis sa traduction. Pour que la traduction soit effective, il faut d'abord créer ce « msgid » dans le fichier « en » - anglais, car c'est le fichier de base pour la traduction dans toutes les autres langues. En créant l'entrée en anglais, et en français, cela fonctionne. Les ajouts seront envoyés à la fin de mon stage, pour une intégration future dans GeoNode. Si GeoNode ne les intègre pas, nous les aurons toujours chez nous en local, ce qui est intéressant.

Traductions via les tags Django

Il existe certaines parties de notre site dont la traduction n'intéresse pas la communauté. Un exemple simple :

« AWARE permet d'accéder aux catalogues de données spatiales du Cirad à La Réunion. »

Traduit en anglais :

« AWARE provides access to spatial data catalogues from the Cirad in Reunion Island. »

Cette traduction ne peut se faire via Transifex, utilisé par toute la communauté. J'ai donc trouvé un autre moyen : dans le code, le langage qu'a choisi l'utilisateur est défini en tant que « LANGUAGE_CODE », qui est une variable dynamique. Ainsi, une simple condition grâce aux tags Django permet d'afficher un certain texte en français ou anglais. Comme nous l'avions évoqué dans l'introduction de cette partie, le choix de la

langue était désactivé pas dans la version 1.0 d'AWARE. En le réactivant, on peut maintenant passer facilement d'une langue à une autre.
La traduction dans toutes les langues n'étant pas une demande de l'équipe, je suis parti du principe que si l'utilisateur est français, le site s'affichera en français. Si l'utilisateur utilise toute autre langue, le site sera affiché dans sa langue (90% du site est traduit par Transifex dans la plupart des autres langues), et le reste (les traductions qui existent uniquement sur notre plateforme) sera traduit en anglais, langue internationale, accessible par la majorité des utilisateurs contrairement au français.
Ce type de traduction est déclenché de cette façon :

```
{% if LANGUAGE_CODE == "fr" %} Bienvenue sur AWARE {% else %} Welcome to AWARE %}{% endif %}
```

Autres traductions

La traduction du site se fait vraiment de multiples manières. Une dernière façon de traduire est d'aller directement dans les fichiers « Static » (extérieurs au répertoire du projet Django). Ceux-ci ne sont pas accessibles pour Django, il faut donc trouver un autre moyen de traduire. Pour GeoExplorer par exemple, il faut se plonger dans le script GeoExplorer.js (fichier de 95 000 lignes lorsqu'il n'est pas minifié) pour ajouter de nouvelles traductions. Ces traductions apparaissent sur le cadre de visualisation des couches et des cartes, et sur la page de création de carte.
Un dernier exemple d'exception dans la traduction, c'est le panier, qui est apparu dans la version 2.4 stable.

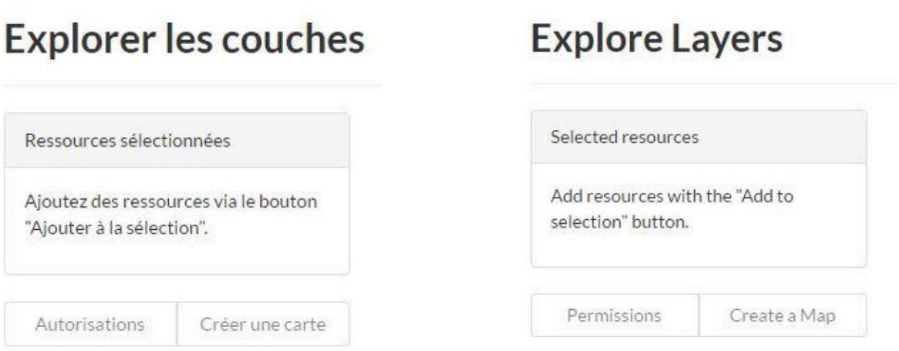


Fig. 24 – Panier traduit en français, et en anglais

Les fichiers relatifs au panier sont à la fois dans « Static » et dans le répertoire Django du projet. Sa traduction n'est pas encore prévue par GeoNode, selon la communauté des développeurs. J'ai donc dû contourner le problème, en modifiant à la fois le fichier HTML du répertoire contenu dans le projet Django, et les fichiers HTML et JavaScript contenus dans le répertoire « Static ».

5. La remontée de notre instance à la communauté GeoNode

La remontée du code sur GitHub (à la communauté, et sur notre profil) devait se faire une fois le développement de toutes les fonctionnalités achevé, et juste avant la mise en production du nouveau site. Cependant, le 15 juin, les développeurs de GeoNode ont imposé une date butoir, invitant tous les contributeurs à envoyer les évolutions qu'ils désireraient voir intégrées dans la version 2.6 avant le 6 juillet, car ils comptaient « geler » la branche Master le 15 juillet.
Nous avons donc décidé de mettre le développement de côté, pour se focaliser sur l'envoi de nos fonctionnalités. Cet envoi est un moyen pour nous de faciliter le passage à une nouvelle version de GeoNode. Si la prochaine version inclut nos spécificités, le passage se fera plus facilement et il y aura moins de conflits entre les branches. C'est également un moyen de faire connaître notre travail à la communauté GeoNode.

a. Premiers pas avec Git et GitHub

Pour contribuer au projet GeoNode, il faut avoir un compte GitHub. GitHub est un service d'hébergement et de gestion de développement de logiciels, qui utilise le logiciel libre de gestion de version Git. C'est ou outil très intéressant en dehors du fait qu'il nous permet de contribuer : notre projet est hébergé, on peut facilement comparer d'anciennes versions de notre projet et y revenir, et surtout il permet d'avoir une excellente traçabilité de toutes les modifications lors d'un projet, grâce à Git.
N'ayant jamais eu l'occasion de pratiquer Git et GitHub, c'était une expérience très enrichissante. Nous avons donc créé un compte sur GitHub. Cette création est gratuite, mais implique en contrepartie que tout utilisateur de GitHub pourra avoir accès gratuitement au code qui sera hébergé.

J'ai créé tout d'abord un *fork*[14] du projet GeoNode original sur notre profil GitHub. J'ai ensuite téléchargé et installé Git sous Windows, puis cloné le fork présent sur GitHub vers Git. Je me suis donc retrouvé avec la branche Master de GeoNode en copie sur ma machine. Cette branche sera la base pour envoyer les fonctionnalités que nous avons développées. Pour chaque fonctionnalité que l'on veut envoyer, il suffit de dupliquer cette branche master, puis de faire toutes les modifications nécessaires pour rendre la fonctionnalité effective, et enfin de faire ce que l'on appelle un *pull request*[15] sur la branche Master du projet GeoNode original (sur le compte GitHub de GeoNode), afin qu'ils l'intègrent, ou non.

L'envoi de nos spécificités s'est fait en deux temps. Dans un premier temps, nous avons commencé à en envoyer sans avoir à installer de nouvelle instance de GeoNode sur notre serveur. Cela a fonctionné au début, mais après une série d'échecs nous avons compris qu'il était nécessaire d'installer une nouvelle version de GeoNode, dédiée à cette tâche.

b. Envoi des premiers *Pull Requests*

L'envoi de nos spécificités a été une tâche assez longue et complexe. Il faut en effet recenser toutes les modifications que nous avons effectuées (que ce soit Hugo Ferrari, Mickaël Mézino ou moi-même) depuis l'installation de GeoNode sur notre serveur, en avril 2015. Je me suis retrouvé face aux mêmes difficultés que j'avais rencontrées lors de la réintégration des modifications effectuées par l'équipe avant mon arrivée en stage. Cette fois-ci, c'est plus compliqué car la branche Master de GeoNode a continué d'évoluer depuis l'installation jusqu'à aujourd'hui. Il faut donc repérer les fichiers modifiés, puis les lignes de codes différentes, sur une quarantaine de fichiers. Cela représente beaucoup de travail car il faut maîtriser et comprendre chaque ligne de code, étudié ensuite par les développeurs de GeoNode, susceptibles de poser des questions.

Pour nos premiers envois de *pull requests*, j'ai installé Git sur Windows, et non sur le serveur. Après avoir cloné la branche Master de GeoNode sur Git, j'ai créé deux branches : une première qui modifie le code pour permettre d'avoir un cadre de visualisation plus grand sur GeoNode, puis une deuxième qui permet de faire un scroll vertical dans la liste déroulante de la barre de recherche. A aucun moment je n'ai eu à tester ces branches, car les modifications étaient minimes (respectivement 2 et 1 fichiers). Je les ai donc envoyées et elles ont été fusionnées à la branche Master de GeoNode sans souci.

Les problèmes sont apparus lorsque j'ai voulu envoyer des branches plus complexes : permettre l'export de métadonnées en format .html et .txt (6 fichiers modifiés), et rendre possible la recherche par catalogue (7 fichiers modifiés). Ces branches impliquent la modification et l'ajout d'un grand nombre de lignes de code, dont beaucoup en Python.
Tout d'abord, GeoNode exige que le code Python respecte la norme PEP8[16], une charte de bonne écriture de code. Le code peut très bien fonctionner, mais si les règles du PEP8 ne sont pas appliquées, la communauté est susceptible de le refuser. Exemple de règles : mettre un espace de chaque côté d'un opérateur, laisser une dernière ligne vide en fin de code, ou encore sauter deux lignes entre chaque

[14] C'est-à-dire un branchement par le clonage d'une branche, ici la branche Master de GeoNode.
[15] Demande de fusion d'une branche spécifique que nous envoyons à la branche Master
[16] Voir https://www.python.org/dev/peps/pep-0008/

fonction – ce sont donc des exigences esthétiques et non fonctionnelles. Il existe un grand nombre de règles, ce qui rend le code difficile à écrire sans faute. Pour résoudre ce problème, j'ai d'abord téléchargé le logiciel *PyCharm* (https://www.jetbrains.com/pycharm/), propriétaire mais gratuit, qui permet d'écrire du code Python en respectant les règles du PEP8. J'ai ensuite simplement utilisé la commande `$ flake8 nom_fichier.py` , qui affiche les erreurs PEP8 directement dans le terminal Linux. Pour pouvoir utiliser cette commande, il faut que le package flake8 soit préalablement installé dans projet.
D'autre part, à la moindre mauvaise indentation dans le code Python, le site peut cesser de fonctionner, mais je ne le vois pas étant donné que je ne peux pas tester ce code sur une plateforme en ligne, je faisais jusque-là les ajouts « à l'aveugle ». Ces branches ne fonctionnaient donc pas quand je les ai envoyées à GeoNode sur GitHub, je m'en suis rendu compte car à chaque *pull request*, un test est effectué : si le test retourne une erreur, le *pull request* est rejeté. Suite à ces échecs, nous avons dû changer de méthode.

c. Echecs, et installation d'une nouvelle instance de GeoNode

Suite à ces échecs, nous avons compris, Mickaël et moi-même, qu'il nous fallait faire des tests en local avant de faire des envois. Or pour faire des tests, il nous a fallu installer une nouvelle instance de GeoNode, de développeur cette fois. Cette tâche – assez pénible – nous a demandé deux jours de travail. Une fois installée, il a été possible de lancer des tests facilement, et de reprendre les envois. Nous avons suite à cela une instance de GeoNode vierge en local, et les fichiers du projet sont dans un répertoire géré par Git, que j'ai réinstallé sur le serveur cette fois. Ainsi, deux clics, nous pouvons changer de branche et tester en direct chacune de celles-ci sur internet.

J'ai alors pu envoyer les deux fonctionnalités que nous n'arrivions pas à envoyer : l'export des métadonnées en .html et en .txt, ainsi que la recherche selon le catalogue.
Restait à envoyer une dernière spécificité assez conséquente impliquant la modification de 29 fichiers, Python et HTML, avec 498 lignes ajoutées et 183 lignes supprimées. Elle comprend deux grands apports :

- Elle permet de rendre visible tout le catalogue, pour chaque utilisateur, même s'il n'est pas connecté, alors que dans GeoNode, si la permission « Voir la ressource » n'est pas donnée, la ressource est invisible. Elle permet également de trier les ressources dans le catalogue selon les permissions : les ressources autorisées pour l'utilisateur en premier (avec un sous-tri par date d'ajout de la ressource), puis les ressources non-autorisées (triées par date d'ajout également). En cliquant sur une ressource non-autorisée, l'utilisateur a accès au téléchargement des métadonnées et au nom de la personne à contacter pour plus d'informations.

- Elle permet d'autre part de limiter les actions des utilisateurs selon les permissions. Dans la version originale de GeoNode, chaque utilisateur connecté peut ajouter des ressources et créer des cartes à partir des ressources du catalogue. Maintenant, cela n'est possible que si l'utilisateur a les droits nécessaires (donnés par l'administrateur ou le propriétaire d'une ressource).

Je comptais rendre ces deux fonctionnalités effectives seulement lorsqu'un flag est passé à « True », spécialement créé dans le fichier de configuration du projet : settings.py. Ce flag n'existe pas dans notre instance, mais comme nous envoyons cette fois nos spécificités à la communauté, il vaut mieux proposer un code qui reste très flexible, car les développeurs sont assez réticents sur le fait d'intégrer un code qui va modifier le fonctionnement de leur instance – et c'est compréhensible. Je pensais ainsi créer un « DEFAULT_SHOW_ALL_CATALOGUE = True », et ensuite créer des conditions « if DEFAULT_SHOW_ALL_CATALOGUE » aux endroits nécessaires dans les 29 fichiers modifiés. Malheureusement je n'ai pas eu le temps d'aller au bout de cette idée, et j'ai donc envoyé le code tel qu'il existe dans notre projet. Finalement, ce *pull request* a été refusé à cause des tests poussés réalisés sur GitHub. Bien que notre instance fonctionne très bien avec ce code, cela crée des conflits lors de la fusion entre notre branche et celle de la branche Master de GeoNode, qui conduisent à un échec des tests. Néanmoins, l'équipe de développeurs de GeoNode a été très intéressée par cette branche, et nous ont demandé de la retravailler afin de rendre son intégration possible.

d. Le cas des applications tierces dans GeoNode

GeoNode intègre des applications tierces. C'est le cas de GeoExplorer, Autocomplete-Light ou encore Django-Taggit.

GeoExplorer est une application web, basée sur le framework GeoExt, qui permet de composer et publier des cartes. Il définit tout ce qui se passe à l'intérieur des cadres de visualisation des couches et des cartes sur GeoNode. Lorsque j'ai intégré le bouton « Zoomer sur l'emprise de la couche », celui-ci existait déjà dans le script GeoExplorer.js, présent dans les fichiers « Static » du projet GeoNode. La traduction des mots affichés dans ce cadre de visualisation ou sur la page de création de cartes sort du contexte de GeoNode, comme nous avons déjà pu le constater. Elle est écrite en dur dans le script de GeoExplorer. Le problème est que les modifications qui touchent à GeoExplorer ne sont pas intégrables au projet GeoNode via des *pull requests* puisqu'il s'agit d'applications tierces. Pour cela, il faut faire des *pull requests* directement sur ces différents projets - présents sur GitHub - puis les modifications seront intégrées lorsque GeoNode mettra à jour la version de ces applications dans son propre projet.

Autocomplete-Light est l'application utilisée par GeoNode pour la barre de recherche. Ainsi, lorsque l'on veut modifier le comportement de la liste déroulante de la barre de recherche, on va chercher les fichiers dans « Static », comme pour GeoExplorer, puis on les modifie. Hugo Ferrari avait modifié le code pour que l'on puisse cliquer directement sur le titre d'une ressource afin d'y être directement redirigé. De ce fait, cette évolution n'a malheureusement pas pu être remontée à la communauté, il faudrait faire un *pull request* directement sur Autocomplete-Light sur GitHub. Seule exception : le scroll dans la liste déroulante de la barre de recherche que j'ai paramétré dans les fichiers CSS du répertoire Autocomplete-Light dans « Static ». Cette fonctionnalité a en effet pu être envoyée au projet GeoNode, qui l'a intégré, car elle est également présente dans le fichier base.less qui comprend un ensemble d'attributs CSS du projet.

Enfin, Django-Taggit est utilisé pour la gestion des mots-clés du projet. Ayant eu des soucis concernant les mots-clés dans GeoNode, j'avais posté un ticket sur le projet sur GitHub, suite à quoi j'ai été invité à le poster directement sur le projet Django-Taggit.

Ces applications ne sont pas les seules applications tierces utilisées par GeoNode, mais ce sont celles dont j'ai eu besoin au cours du projet, et que j'ai modifiées. Il en existe beaucoup d'autres.

V. Bilan

1. Les nouvelles fonctionnalités

a. Les fonctionnalités validées

A la fin de mon stage, la plupart des évolutions qui m'avaient été assignées ont été réalisées. Parmi celles-ci, certaines étaient prévues dès le début, d'autres m'ont été assignées durant le stage :

Evolutions prévues dès le début du stage :

- Passer à la version GeoNode 2.4 stable si cela présente un réel intérêt, et n'entraîne pas de régressions,
- Améliorer la visibilité sur le site, en permettant de ne pas faire apparaître la ressource lorsqu'on vient de la publier, si besoin,
- Améliorer la prise en compte des permissions, en masquant certains boutons permettant de visualiser des ressources ou d'éditer/contribuer si l'utilisateur n'y est pas autorisé,
- Compléter la traduction d'AWARE,
- Ajouter une section « Derniers ajouts »,
- Récupérer l'adresse mail de l'utilisateur depuis l'annuaire LDAP lors de sa première connexion,
- Résoudre le bug présent sur la version d'AWARE en production à mon arrivée en stage, qui ajoute un utilisateur de façon automatique et non désirée lorsqu'on définit les permissions,
- Agrandir le cadre de visualisation des ressources,
- Rendre la légende apparente lors de la visualisation d'une couche,
- Rajouter les logos des partenaires en bas de page d'accueil,
- Faire défiler le carrousel présent sur la page d'accueil automatiquement.

Evolutions assignées durant le stage :

- Permettre à l'administrateur d'ajouter des ressources et de modifier les permissions sans qu'il ait besoin de s'intégrer lui-même dans un groupe « admin »,
- Ajouter un bouton « Zoomer sur l'emprise de la couche » sur le cadre de visualisation des couches,
- Permettre le scroll dans la liste déroulante de la barre de recherche,
- Changer l'affichage du nom des utilisateurs,
- Rendre la saisie de la « Région » obligatoire lors de la saisie des métadonnées.

J'ai également effectué de légères modifications afin d'améliorer l'ergonomie et l'apparence de la plateforme : en modifiant l'apparence et l'appellation du « panier », apparu avec la migration vers GeoNode 2.4 stable ; en ajoutant un bouton de pagination en bas de page, qui évite de devoir remonter en haut de page dès que l'on veut accéder à la page suivante ou précédente ; en ajoutant un logo de type « checked » lorsqu'un filtre est sélectionné.

Les évolutions qui ont été apportées durant ce stage sont en accord avec celles définies par l'équipe à mon arrivée, et durant le stage. Certaines seront toujours à améliorer, je pense notamment à la traduction du site, qui est à retravailler à chaque ajout de texte spécifique à AWARE ou à chaque migration vers une nouvelle version.

Les fonctionnalités qui n'ont pas pu être développées

Certaines spécificités n'ont malheureusement pas pu être développées. Certaines par manque de temps dans le délai de mon stage, d'autres dépassant, je pense, légèrement le cadre de mes compétences en informatique (je pense notamment au passage en SSL).

Effectuer un contrôle sur la saisie des mots-clés

Sur la version d'AWARE en production, il y a un problème lorsqu'un utilisateur ajoute des mots-clés. En effet, si le dernier (ou l'unique) mot-clé est un mot composé (exemple : Limites administratives), le mot est systématiquement coupé, donnant « Limites » et « administratives ». Pour pallier à cela, il faut : soit mettre le mot composé entre guillemets (ce qui est fait automatiquement par GeoNode lorsqu'on a une liste de mots-clés, sauf sur le dernier), soit terminer la liste par une virgule, ce qui est contraignant pour l'utilisateur.
A ce jour je n'ai pas trouvé la solution à ce problème, mais quelques échanges avec les développeurs de GeoNode me font penser que celle-ci se trouve dans l'application tierce « Django-Taggit » que nous avons évoquée plus tôt.

Import des fichiers SLD

L'import de fichiers « .sld » est une fonctionnalité que je n'ai pas pu améliorer. L'import en soi n'est pas une fonctionnalité à développer, à proprement parler, puisque GeoServer le permet mais en version 1.0. Or, les fichiers SLD issus de QGIS par exemple sont en version 1.1. Sur la version 2.4 stable de GeoNode, nous utilisons GeoServer 2.7, mais ni sur la version 2.9, ni sur la version 2.10 de GeoServer les imports de fichiers SLD en 1.1 ne sont prévus. Il faudra donc sûrement attendre encore, à moins de trouver une autre solution.
Nous notons tout de même que depuis le passage à GeoNode 2.4 stable, nous bénéficions de quelques évolutions dues à GeoServer 2.7. Avant, rien n'était pris en compte concernant le style des fichiers. Maintenant, les classes du .sld sont prises en compte, et il ne reste plus qu'à appliquer le bon code couleur pour retrouver le rendu original, ce qui représente un gain de temps important.

Passer en SSL

Passer en SSL (https) s'est avéré compliqué. Après trois essais différents, nous ne sommes pas parvenus, avec Mickaël Mézino, à effectuer la transition.

Tout d'abord, il nous fallait un certificat. Nous avons d'abord testé un certificat auto-signé, mais cela peut provoquer le blocage du site en raison d'exceptions de sécurité. Le navigateur va informer l'utilisateur que le certificat du site est peu fiable. En plus de cela, il sera mal perçu par Google. Dans ce cas, autant garder un site en http sans notification du navigateur.

Nous avons donc obtenu un certificat validé par une autorité du web, ce qui nous a permis de nous pencher ensuite sur le code. Pour effectuer cette transition, nous avons suivi la documentation de GeoNode « Running GeoNode under SSL »[17]. Cependant, en suivant les instructions à la lettre, cela ne fonctionne pas. Nous avons repéré des modifications à effectuer – qui ne sont pas indiquées dans la documentation – sans lesquelles le site ne fonctionne plus du tout. Lors de notre dernière tentative, le site fonctionnait presque partout, sauf lors de la modification de ressources : le serveur nous retournait une « Handshake Error ». D'autres erreurs apparaissaient : sur certaines pages le serveur continuait à aller chercher des pages en « http » au lieu du « https ». Enfin, sur certaines pages, le serveur créait de mauvaises URLs, oubliant un « / » après le nom du domaine, compromettant l'accès à la page.

Finalement, après avoir contacté l'équipe de développeurs de GeoNode, il semblerait que le site sous SSL n'ait jamais vraiment été testé, et que la documentation présente sur le site soit en réalité une liste d'étapes théoriques. Cependant, faisant partie de la « mailing list » de GeoNode, nous avons pu constater que des développeurs étaient actuellement en train de travailler dessus. Il sera donc peut-être bientôt possible d'effectuer la transition.

[17] Voir http://docs.geonode.org/en/master/tutorials/advanced/geonode_production/ssl.html

Traduction des Notifications

En fin de stage, je me suis rendu compte que la traduction de la section « Notifications » ne fonctionnait pas en français. Pourtant, en se penchant sur le code, toutes les traductions existent et sont prévues par Transifex. J'ai donc posté un ticket[18] pour signaler un problème sur le projet GeoNode sur GitHub, et des développeurs de la communauté sont actuellement en train de l'étudier.

2. La remontée de notre version à GeoNode

a. Un certain nombre de spécificités intégrées à la branche Master de GeoNode

En fin de stage, certaines fonctionnalités ont bien été intégrées à la branche Master de GeoNode. L'équipe de développeurs a intégré l'agrandissement de la taille du cadre de visualisation des couches et des cartes, le scroll dans la liste déroulante de la barre de recherche, la recherche en fonction du catalogue dans lequel on se trouve, et l'export des métadonnées en .html et .txt. Reste à savoir si toutes ces fonctionnalités feront bien partie de la version 2.6 actuellement en développement.

Les fonctionnalités les plus lourdes n'ont pas été intégrées à ce jour, car elles demandent un travail conséquent d'adaptation au code original de GeoNode. Le codeproposé doit passer une série de tests poussés, qui n'aboutissent pas forcément. Quoi qu'il en soit, le fait que la communauté ait intégré les fonctionnalités énoncées précédemment entrainera une charge de travail moins importante si l'équipe ARTISTS décide de migrer vers la version 2.6. De plus, le fait de faire des *pull requests* vers GeoNode m'a obligé à créer des branches spécifiques pour chaque fonctionnalité, qui sont maintenant présentes sur notre profil GitHub. En cas de migration vers la version 2.6, l'équipe pourra intégrer ces fonctionnalités plus facilement en effectuant des fusions (*merge*) depuis GitHub, option plus simple que de copier le code manuellement. GitHub offre la possibilité de différencier plusieurs branches, mettant en évidence les modifications sans recourir au *merge*.

b. Quelles perspectives pour la diffusion de notre instance de GeoNode ?

Le dernier *pull request* a été plus compliqué à effectuer que les autres, puisqu'il comportait de profonds changements. Il a bien été envoyé, mais les tests n'ont pas tous abouti, or pour qu'un *pull request* soit validé, il faut que tous ces tests aboutissent. La raison de cet échec est liée à la modification de fichiers sensibles, relatifs à la sécurité. Plus tard, il serait envisageable de passer en revue ces modifications pour les ajuster légèrement afin qu'ils passent les tests. Je n'ai malheureusement pas pu m'en occuper en fin de stage, car cela nécessite un certain temps, sans certitude d'y aboutir.

Il serait également envisageable, à l'avenir, de modifier notre code afin de le rendre beaucoup plus souple. En l'état, la branche que nous avons envoyée modifie le comportement de GeoNode, mais il serait possible que le code original de GeoNode puisse intégrer le nôtre, sans que l'un vienne perturber l'autre. Pour cela, il faudrait créer – comme nous l'avons évoqué précédemment – des paramètres supplémentaires dans le fichier settings.py. J'ai tenté de créer un flag, qui fonctionnait en partie, mais je n'ai pas eu le temps d'aller jusqu'au bout de cette idée, qui demande beaucoup de travail. En l'état, un autre problème est que pour obtenir le comportement d'AWARE, il faut activer un flag « SKIP_PERMS_FILTER = True » déjà existant sur GeoNode, ce qui peut avoir des effets de bord sur une instance différente de la nôtre.

3. Bilan personnel

Ce projet a vraiment été très formateur. C'était tout ce que j'attendais de mon stage de fin d'études. Etant donné que j'avais déjà de bonnes bases en géomatique, je souhaitais profiter de ce stage pour développer plus de compétences en webmapping et en programmation, et il m'en a donné

[18] Voir https://github.com/GeoNode/geonode/issues/2575

l'opportunité. N'ayant pas de formation purement informatique, j'ai vraiment apprécié le fait de pouvoir développer dans le cadre d'un projet au sein de cette équipe, qui m'a laissé beaucoup de libertés, et qui a toujours été là pour me guider et pour m'aider. J'ai appris beaucoup de choses, en passant par des moments parfois difficiles. Je commencerai ici par aborder les principales difficultés que j'ai rencontrées, pour ensuite me pencher sur les compétences que ce stage m'a permis de développer.

a. Les difficultés rencontrées

Durant ce projet, je me suis parfois retrouvé en difficulté. Mais plus le stage avançait, plus je gagnais en assurance, me permettant de les surmonter avec plus d'aisance. Les connaissances que j'ai assimilées en cours de route m'ont toujours servies par la suite. Des connexions se sont faites entre ce que j'ai appris dans les différents langages.

Tout d'abord, concernant l'environnement de travail. Étant habitué à travailler sous Windows, il a fallu m'habituer au terminal Linux. C'est au début assez déroutant – même si nous avions eu l'occasion de travailler sous ce système cette année – mais je m'y suis vite fait, pour finalement trouver cela extrêmement utile et pratique.

De la même manière, il était difficile pour moi de conceptualiser l'architecture des systèmes utilisés, car c'est une chose assez abstraite. En début de stage, la « DMZ » ou faire un « clone du serveur en production » étaient des termes qui ne me parlaient pas vraiment.

Parfois le développement a pu être vraiment déroutant, et déstabilisant. Au début de mon stage, il était difficile de m'y retrouver dans le grand nombre de fichiers présents dans le projet, et je ne pouvais même pas trier ces fichiers selon leur importance dans le projet. Un projet GeoNode compte à peu près 2200 fichiers au total, mais avec le temps, j'ai compris qu'à peine 100 fichiers étaient vraiment à modifier dans le projet, et une trentaine de fichiers étaient vraiment clés.

Un des aspects très déstabilisant, que ce soit dans le répertoire du projet, ou sur les documentations en ligne, est d'arriver à trouver ce qui m'intéresse, et ce que je cherche dans cette masse d'information. De même, lorsqu'une erreur se produit et que plus rien ne fonctionne, je me disais au début que l'erreur pouvait vraiment venir de n'importe où. Finalement, grâce au terminal Linux, il existe de multiples manières de trouver exactement ce que l'on cherche, il devient plus simple de parcourir le répertoire du projet, de faire ressortir les erreurs et de trouver quelle en est l'origine.

b. Ce que j'ai appris

Ce stage m'a permis de vraiment développer mes compétences, à la fois en programmation web, en programmation plus générale, en webmapping, en informatique ainsi qu'en gestion de projet. J'ai bien évidemment pu mettre à profit mes compétences en SIG, qui m'ont servi tout au long de mon stage.
Je connaissais déjà le langage Python, mais je n'avais jamais eu l'occasion de le pratiquer autant. De même pour les langages HTML et CSS. J'ai appris à mieux maîtriser le langage JavaScript, ainsi que la bibliothèque JQuery (framework de JavaScript). J'ai également découvert AngularJS, autre framework de JavaScript dont je ne connaissais pas l'existence. Ce stage a enfin été l'occasion de découvrir le framework Django, que j'ai trouvé assez intuitif, et dont la documentation en ligne est vraiment complète.

J'ai pu progresser à la fois en travaillant directement sur la plateforme, en consultant des forums de programmation, en observant les techniques de mes encadrants, Agnès Tendero et Mickaël Mézino, et en consultant des cours en ligne sur les différents langages de programmation utilisés.
Durant ce stage, j'ai également commencé à assimiler tout un langage propre à l'informatique et à l'administration système, qui était assez nouveau pour moi. Ayant débuté ce stage avec un peu d'appréhension, je suis finalement très satisfait de ce que j'ai appris, et de ce que j'ai réalisé.

4. Perspectives

En six mois de stage, il n'est pas possible de tout réaliser. De plus, certains objectifs viennent se greffer en cours de route. Ainsi, certaines tâches restent à effectuer, et voici donc les axes sur lesquels j'aurais continué à travailler si mon stage avait été plus long.

Tout d'abord j'aurais continué à améliorer la traduction d'AWARE, mais aussi de GeoNode via Transifex. A chaque fois que la communauté GeoNode décide d'intégrer certaines de nos fonctionnalités, ce sont autant de nouveaux termes à ajouter dans Transifex pour qu'ils soient ensuite traduits dans toutes les langues.
J'aurais également pris plus de temps pour effectuer les *pull requests*. Comme je l'avais évoqué plus tôt, il faut rester le plus flexible possible lorsque l'on contribue.
Dans les semaines et les mois qui arrivent, il serait intéressant de savoir si les développeurs parviennent à faire fonctionner le SSL sur une instance de GeoNode, ce qui nous permettrait faire la transition à notre tour.
De la même manière, il faudrait surveiller les prochaines améliorations prévues sur GeoNode, pour voir si une version plus récente des fichiers .sld sera supportée prochainement.
Si j'étais resté quelques semaines de plus en sein de l'équipe, j'aurais pu effectuer la migration vers GeoNode 2.6 qui va sortir très prochainement, et réintégrer nos spécificités en mettant à profit ce que j'ai appris des réintégrations que j'ai effectuées en début de stage. Je pense que j'aurais cette fois pu le faire assez rapidement.
Enfin, j'aurais pu mettre en place le bandeau qui informe les utilisateurs que la plateforme utilise des cookies, ce que la loi impose désormais aux responsables de sites[19].

[19] Voir https://www.cnil.fr/fr/site-web-cookies-et-autres-traceurs

VI. Conclusion

Aujourd'hui, l'Atlas cartographique web AWARE est une réelle plateforme à part entière, avec un fonctionnement très personnalisé. Elle est le fruit d'un travail d'équipe, et de nombreuses heures de réflexion, de travail, d'échecs et de succès. Personnellement, ce stage de fin de Master 2 m'a apporté beaucoup de satisfaction. Il représente tout ce que j'attendais de mon stage de fin d'études, dernière étape avant l'entrée dans le monde professionnel.

A l'issue de ce stage, le cahier des charges fixé par mes encadrants a été respecté, même s'il reste encore quelques axes à développer. L'équipe ARTISTS désirait améliorer la confidentialité sur la plateforme AWARE, et des solutions ont été trouvées afin de répondre à cet objectif. La migration vers la version 2.4 stable a été assurée, ce qui a permis de résoudre certaines anomalies présentes sur la plateforme d'une part, et d'apporter de nouvelles fonctionnalités d'autre part. De nombreuses évolutions ont été apportées durant le stage, certaines prévues en début de stage et d'autres qui sont apparues en cours de route. Finalement, la plateforme est prête pour la mise en production à la fin de mon stage, et il est vraiment gratifiant de voir les fruits de son travail accessibles en ligne.

N'étant pas de formation purement informatique, ce stage représentait un réel challenge pour moi. Je craignais parfois de ne pas être à la hauteur des exigences demandées, ou de ne pas avoir les compétences suffisantes pour réaliser les différentes missions. Finalement, ce stage m'a montré qu'en le voulant vraiment, et en s'investissant, il était possible de faire du développement web, et d'y prendre du plaisir. Je me suis souvent retrouvé bloqué en début de stage, mais plus celui-ci avançait, plus je trouvais des solutions facilement, me permettant d'aller toujours plus loin dans le développement de la plateforme. Je remercie l'équipe ARTISTS d'avoir fait le pari de prendre un étudiant non-spécialiste en informatique, et j'espère avoir pu leur apporter en retour.

Ce stage me conforte dans mon choix d'études en géomatique, et il m'a vraiment donné envie d'aller plus loin dans ce domaine, plus particulièrement dans le développement web, le webmapping, la programmation en général, mais aussi la gestion de bases de données.

VII. Lexique

API : Application Programming Interface. Ensemble de classes, méthodes et fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

DMZ : « Zone Démilitarisée ». En informatique, il désigne un sous-réseau, séparé du réseau local et d'internet par un pare-feu. Il protège le réseau local des accès via internet, et les services susceptibles d'être accédés depuis internet sont situés en DMZ.

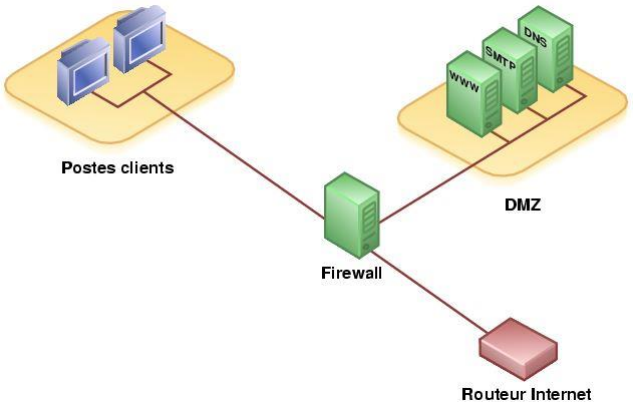


Fig. 25 – Schéma réseau d'une utilisation de DMZ avec un pare-feu. Source : Wikipédia

Framework : Littéralement, « Cadre de travail ». Ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations d'un logiciel. Il se distingue par son caractère générique, non spécialisé, mais qui peut faire appel à des bibliothèques spécialisées. C'est donc un cadre de travail : il guide le développeur, et l'invite à respecter des modèles préalablement créés.

LDAP : Lightweight Directory Access Protocol, ou Protocole d'accès aux annuaires léger en français. Il s'agit d'un protocole permettant l'interrogation et la modification des services d'annuaire.

Pull request : Méthode pour soumettre une contribution à un projet open source.

SDI (**IDS** en français) : Infrastructure de données spatiales. Organisation qui repose sur des accords de partage, une coordination entre ses membres et des systèmes informatiques qui intègrent un ensemble de services (catalogues, serveurs, logiciels, données, pages web, etc.) utilisés pour la gestion de l'information géographique.

SLD (fichier) : Descripteur de Style de Couche. Schéma XML indiqué par l'OGC pour décrire le style des couches de carte. Il est capable de traiter des données vectorielles et Raster.

SSH : Secure SHell. Protocole de communication sécurisé. Il impose un échange de clés de chiffrement en début de connexion. Ensuite, tous les segments TCP (Protocole de Transport) sont authentifiés et chiffrés, et il devient impossible pour quiconque de voir ce que fait l'utilisateur et d'intercepter des données qui transitent.

SSL : Secure Sockets Layer. Protocole de sécurisation des échanges sur internet. La session est chiffrée, ce qui empêche un tiers d'intercepter des données sensibles transitant par le réseau. Le SSL est surtout utilisé pour les sites proposant des achats en ligne. On parle maintenant de TLS (Transport Layer Security), son successeur.

Template : Patron ou gabarit de mise en page permettant de garder une certaine cohérence entre les différentes pages d'un site web.

Thumbnail : Image miniature d'une ressource permettant d'en avoir un aperçu.

VIII. Bibliographie

Ferrari, H. (2015). *Mise en place d'un atlas cartographique web de données agro-environnementales interopérable basé sur des technologies libres*, Mémoire, Université Toulouse Jean-Jaurès.

http://www.cirad.fr/

http://reunion-mayotte.cirad.fr/

https://docs.djangoproject.com/

http://docs.geonode.org/en/master/

https://github.com/

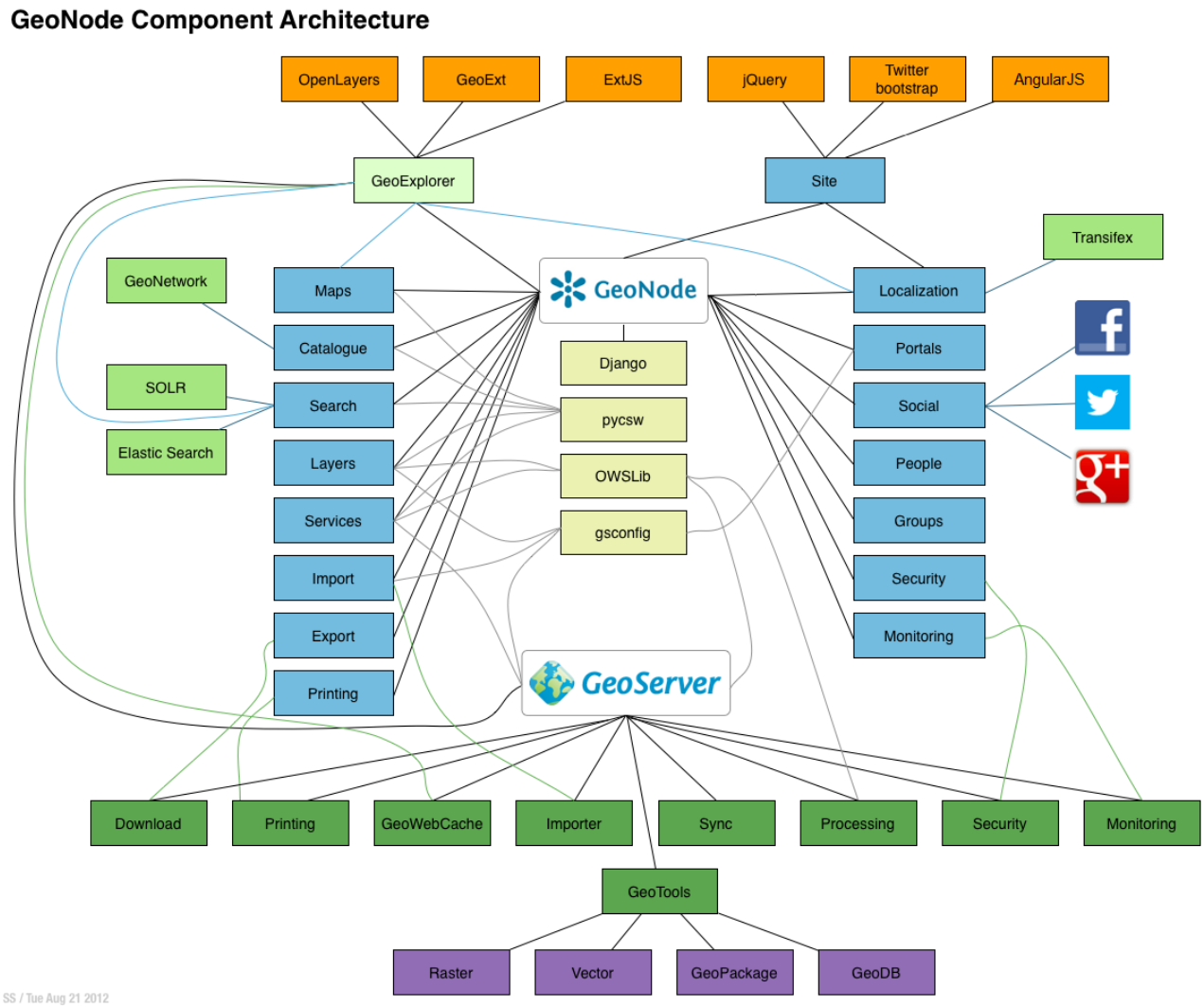
https://www.transifex.com/

http://stackoverflow.com/

https://openclassrooms.com/

https://fr.wikipedia.org/

IX. Annexes



Annexe I - Communication entre les briques logicielles pour un projet GeoNode

Hello,

Here is a first list of the functionnalities we've developped :

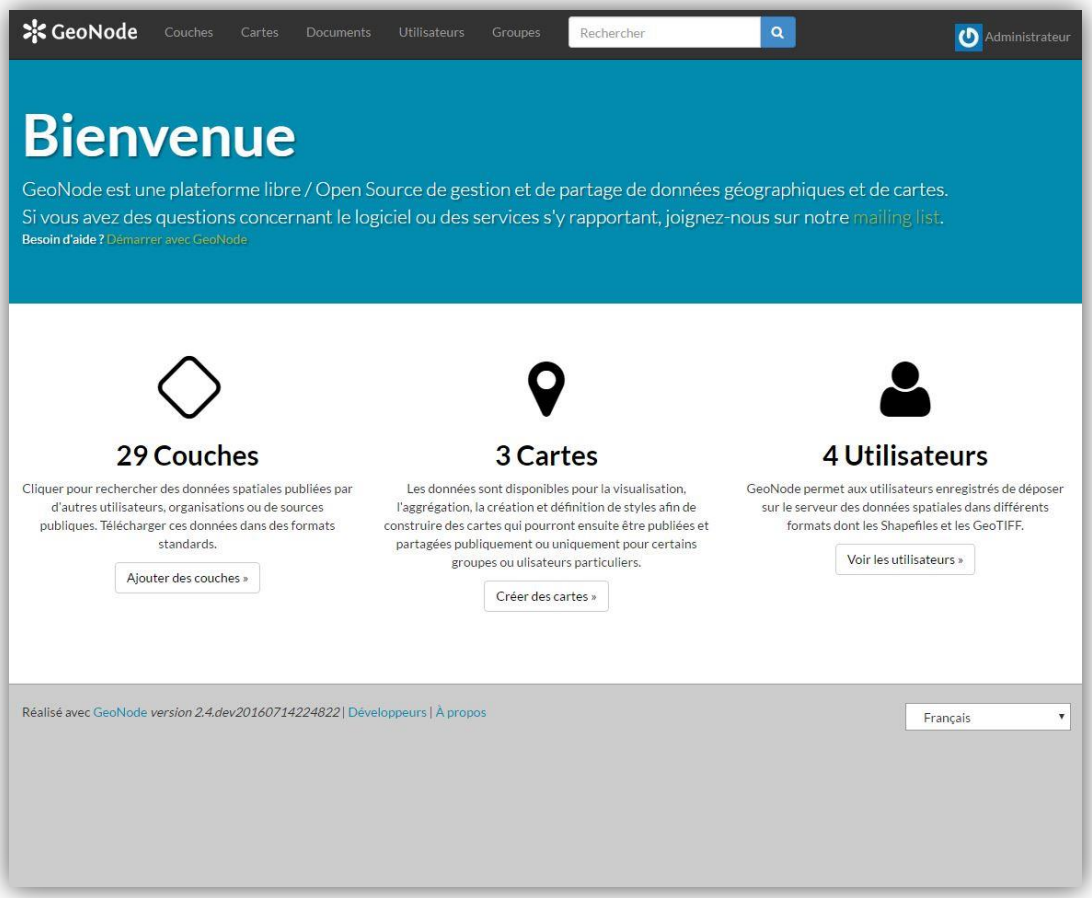
1. Concatenating the metadata Regions in Title search fields
2. Display the full list of resources of each catalogue for every user (logged in or not)
3. List of layers/maps/documents sorted by permission : first the authorized ones, then the restricted ones.
4.. Export metadata to text and HTML format
5. Restrictions for unauthenticated users : hide 'Add layer', 'Create new map', 'Add documents' options
6. Search by text (title) improved within each catalogue : search did not take into account if users are in the layers catalogue and could propose maps and documents.
7. Direct access to the layer/map/document when the Title is selected int the search fields
8. Notifications : users can manage their notifications in the menu
9. Add a button to zoom on layer extent in the geoexplorer toolbar
10. Height of the geoexplorer frame increased in the layer/map sections
11. Display/hide functionnalities depending on permissions (view resource base, add resource base), see https://lists.osgeo.org/pipermail/geonode-users/2015-June/thread.html

I can give you more details if you want such as modified files names,...

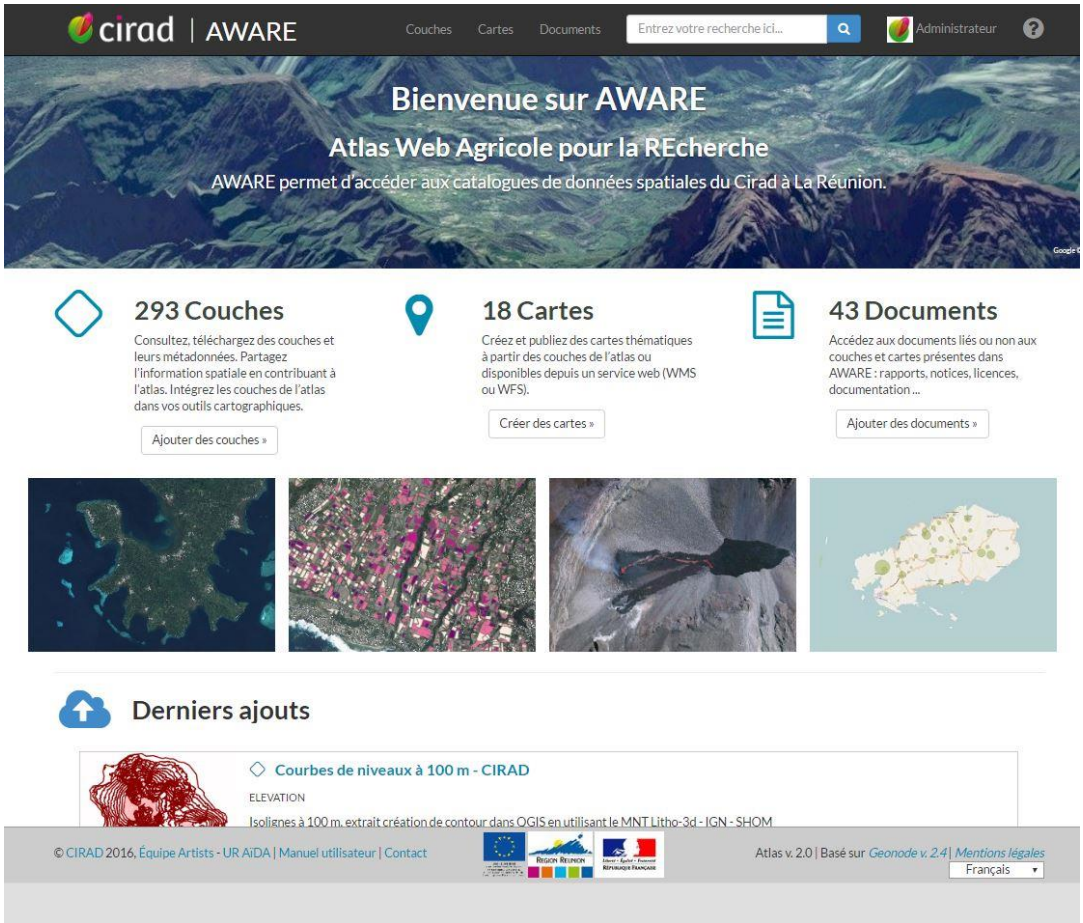
And I'll send you soon the others functionnalities list.

Agnès Tendero

Annexe II – Mail envoyé à l'équipe de développeurs de GeoNode, leur proposant nos spécificités



Annexe III – Page d'accueil lors de l'installation de GeoNode 2.4 stable en avril 2016



Annexe IV – Page d'accueil avant la mise en production en août 2016